



The RHIZOME architecture: a hybrid neurobehavioral control architecture for autonomous vision-based indoor robot navigation

Dalia Marcela Rojas Castro

► To cite this version:

Dalia Marcela Rojas Castro. The RHIZOME architecture: a hybrid neurobehavioral control architecture for autonomous vision-based indoor robot navigation. Robotics [cs.RO]. Université de La Rochelle, 2017. English. NNT: 2017LAROS001 . tel-01753804

HAL Id: tel-01753804

<https://theses.hal.science/tel-01753804>

Submitted on 29 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



UNIVERSITÉ DE LA ROCHELLE

ÉCOLE DOCTORALE S2IM

LABORATOIRE : L3I

THÈSE présentée le 11 janvier 2017 par :

Dalia Marcela ROJAS CASTRO

pour obtenir le grade de : **Docteur de l'université de La Rochelle**

Discipline : **Informatique et Applications**

The RHIZOME Architecture : A Hybrid Neurobehavioral Control Architecture for Autonomous Vision-based Indoor Robot Navigation

RAPPORTEURS	Patrick HÉNAFF Olivier STRAUSS	Professeur des Universités, LORIA, École de Mines de Nancy Maître de conférences HDR, LIRMM, Université de Montpellier
EXAMINATEURS	Philippe GAUSSIER Jean-Philippe DOMENGER Armelle PRIGENT Emmanuelle GRANGIER	Professeur des Universités, ETIS, Université de Cergy-Pontoise Professeur des Universités, LABRI, Université Bordeaux 1 Maître de Conférences, L3i, Université de La Rochelle Docteur et Enseignante, UFR Intermédia, Université de Toulon
DIRECTION	Michel MÉNARD Arnaud REVEL	Professeur des Universités, L3i, Université de La Rochelle Professeur des Universités, L3i, Université de La Rochelle



*A mi madre por darme alas e impulso para volar muy alto y aun así, ser el nido que me
abriga en momentos de tormenta.*

A Claudio por el apoyo incondicional que me ha permitido alcanzar mis sueños.

Credits for this journey

The dissertation here presented, is the result of a thorough research work I have accomplished during a journey I started four years ago.

As travel-lover, most of my journeys have been driven by the desire of the discovery of the unknown: new places, new people, new cultures etc., representing all, new worlds to be discovered. This constant longing of nourishing my curiosity and learning, has led me towards many extraordinary paths and more particularly towards the path of getting and accomplishing a Ph.D. Certainly, it has not been the easiest and the smoothest path, but it has been one of the most challenging and fulfilling adventures I have ever experienced.

Through the years of this journey, I had the great opportunity to meet and spend time with many marvelous people without whom the achievement of this work would have not been possible. They have helped me in a way or another, at one time or another to go through this path and achieve it successfully. Hence, the following words are an attempt to express, in the best way I can, my deep gratitude to all of you. I call it an attempt, not because of the effort of expressing it through a collection of meaningful words but because of the impossibility of it being up to the level of the actions, gestures, presence, time and words, each of you offered me when I most needed it.

First and foremost, I would like to thank my thesis advisors, Pr. Dr. Arnaud Revel and Pr. Dr. Michel Ménard for your valuable guidance, expertise and support. My discussions with both of you throughout this whole journey were always interesting and inspiring. Your wisdom and experience in the research world helped me ensure that I followed a proper line of reasoning to reach the generated results. Your strong belief in the potential of this research and the trust you have granted me has been a tremendous influence towards the completion of this work. For all that thank you very much.

I also would like to thank Pr. Dr. Patrick Hénnaï and Dr. Olivier Strauss for having reviewed my dissertation and for sharing interesting comments. I am also grateful to Pr. Dr. Jean-Philippe Domenger, Dr. Armelle Prigent and Dr. Emmanuelle Grangier for having accepted to assess my thesis by being part of the jury and Pr. Dr. Philippe Gaussier to have presided the jury. I would like to thank all contributors of the Poitou-Charentes region (France) and the European Regional development fund for their generous financial support.

My sincere gratitude also goes to Pr. Dr. Jean-Marc Ogier (director of the laboratory L3i until 2016) for providing me all the facilities to carry out this work in a great environment and atmosphere. Special thanks to those who trusted in my capacity as a teacher and showed me the best example of how to teach, Dr. Frederic Bertrand, Dr. Vincent Courbouley, Nicolas Trougeon, Dr. Christophe Demko, Dr. Mickael Coustaty and Dr. Renaud Peteri.

Special thanks to the all the supportive and wonderful colleagues of the laboratory L3i and for their constant help and kindness, Kathy, Erlandri, Genevieve, Dominique, Romain, Mickael, Joseph, Muzzamil, Jane, Maroua, Poh Lee, Stephane ... and all the rest.

My heartfelt gratitude to all my friends that by his or her presence in my life, made of this a memorable time:

Sahil, despite the long distance, you traveled along my side during the first years of this journey. You were always eager to help me through the toughest challenges I encountered and I am grateful for that. Thanks for proofreading my very first articles and the introduction of this dissertation. Your good tips were always helpful to improve further my writing in English.

A mes amis de tous les jours pour les échanges scientifiques, les réflexions stimulantes, les discussions superflues et profondes, les danses latinos arythmiques, les balades à vélos, les baignades dans la mer et dans la piscine, les raclettes en hiver et barbecues en été... à vous tous, merci beaucoup pour ces années magiques, Audrey, Bich, Chris, Clément, Chloé, Cyril, Damien, Daouda, Dounia, Elo, Imen, Ghina, Guillaume, Nam, Pohoung, Rouaa, Romain, Sebastien et Vincent.

A mon meilleur ami, Romanito. Quand je me suis embarquée dans cette aventure, tu y étais déjà et tu m'as accompagnée jusqu'à la fin. Nos longues conversations sur la vie, nos débats sur le monde académique, nos discussions scientifiques, les heures passées à comprendre ma thèse, mon initiation à la guitare, ta passion pour la musique... et tant d'autres moments partagés, ont été pour moi des sources d'inspiration, de motivation et d'apprentissage constant. Merci beaucoup d'avoir rendu ce voyage plus agréable par ta présence.

Une spéciale dédicace et ma profonde gratitude à toi Vincent pour cette dernière et difficile année de thèse. Ton soutien au quotidien a été d'une grande aide pour l'aboutissement de cette thèse toute en gardant une (quasi) bonne santé mentale. Ta présence et ton assistance durant les moments difficiles ont été sans faille et je te remercie de tout mon cœur. Je profite aussi pour vous remercier également Marie et Dominique pour votre aide dans les moments de difficulté.

A mis amigas y amigos de infancia y de tantos años, muchas gracias por su linda amistad. Gracias por alegrarse de mis triunfos y acompañarme en mis derrotas.

Tiago, el menino brasileiro. Muchas gracias por todos los momentos compartidos. Lograste ofrecerme los mejores momentos de evasión latina, oxígeno necesario para sobrevivir en este arduo viaje que ha sido el doctorado.

Luisfer, muchas gracias a ti por tu apoyo incondicional durante tantos años de amistad. Tus sabias palabras fueron siempre precisas y claves para el desarrollo de mi trabajo tanto profesional como personal.

Last but not least, my deepest gratitude goes to you my dear family:

Quiero agradecer de corazón a todos los miembros de mi familia. Ustedes me han dado siempre las fuerzas necesarias para continuar el camino. Porque a pesar de la distancia y del tiempo, siempre han estado cerquita de mí apoyándome, aconsejándome y enviándome las mejores energías para avanzar. Muchas gracias por todo su cariño.

Camilo, mio caro fratellino. Sentirti vicino a me con la tua nuova famiglia è stato un bel regalo di vita. Grazie a Sarita, Daniela, Gianni e Nonna per avermi accolte sempre nella vostra casa come se fossi della vostra famiglia. Ogni viaggio da voi, ogni bella cena nella campagna Italiana, mi ha permesso di prendere le forze quando non ce l'avevo più e di ritornare nuova per continuare il lavoro. Grazie mille per tutti. Siete formidabili.

Carito y Esteban, gracias por su compañía, alegrías, abrazos, conversaciones, juegos y más durante su estadía en Francia. Tenerlos aquí a mi lado fue muy lindo y aunque no fue fácil para ninguno de nosotros por la novedad de la situación, fue el mejor de los años vividos. Gracias por todas esas experiencias compartidas.

A mi mamá amada, gracias por todo, por la vida que me has dado, por tu amor y apoyo incondicional que me ha permitido lograr hasta ahora todos mis objetivos. Gracias por creer siempre en mí, por ser el mejor ejemplo de fortaleza, perseverancia y verriquera para alcanzar lo inalcanzable.

Abstract

The RHIZOME Architecture: A Hybrid Neurobehavioral Control Architecture for Autonomous Vision-based Indoor Robot Navigation

The work described in this dissertation is a contribution to the problem of autonomous indoor vision-based mobile robot navigation, which is still a vast ongoing research topic. It addresses it by trying to conciliate all differences found among the state-of-the-art control architecture paradigms and navigation strategies. Hence, the author proposes the RHIZOME architecture (**R**obotic **H**ybrid **I**ndoor-**Z**one **O**perational **M**odule): a robotic control architecture capable of creating a synergy of different approaches by merging them into a neural system. The interactions of the robot with its environment and the multiple neural connections allow the whole system to adapt to navigation conditions.

The RHIZOME architecture preserves all the advantages of behavior-based architectures such as rapid responses to unforeseen problems in dynamic environments while combining it with the *a priori* knowledge of the world used in deliberative architectures. However, this knowledge is used to only corroborate the dynamic visual perception information and embedded knowledge, instead of directly controlling the actions of the robot as most hybrid architectures do. The information is represented by a sequence of artificial navigation signs leading to the final destination that are expected to be found in the navigation path. Such sequence is provided to the robot either by means of a program command or by enabling it to extract itself the sequence from a floor plan. This latter implies the execution of a floor plan analysis process. Consequently, in order to take the right decision during navigation, the robot processes both set of information, compares them in real time and reacts accordingly. When navigation signs are not present in the navigation environment as expected, the RHIZOME architecture builds new reference places from landmark constellations, which are extracted from these places and learns them. Thus, during navigation, the robot can use this new information to achieve its final destination by overcoming unforeseen situations.

The overall architecture has been implemented on the NAO humanoid robot. Real-time experimental results during indoor navigation under both, deterministic and stochastic scenarios, show the feasibility and robustness of the proposed unified approach.

Keywords: Artificial neuronal network-based control architecture, autonomous mobile robot indoor navigation, visual perception, data merging, floor plan analysis, pattern recognition, behavior-based hybrid approach.

Résumé

L'architecture RHIZOME: Une Architecture de Contrôle Neurocomportementale Hybride pour la Navigation Autonome Indoor des Robots Mobiles Reposant sur la Perception Visuelle

Les travaux décrits dans cette thèse apportent une contribution au problème de la navigation autonome de robots mobiles dans un contexte de vision indoor. Il s'agit de chercher à concilier les avantages des différents paradigmes d'architecture de contrôle et des stratégies de navigation. Nous proposons dans ce but l'architecture RHIZOME (**R**obotic **H**ybrid **I**ndoor-**Z**one **O**perational **M**odule): une architecture de contrôle robotique mettant en synergie ces différentes approches en s'appuyant sur un système neuronale. Les interactions du robot avec son environnement ainsi que les multiples connexions neuronales permettent à l'ensemble du système de s'adapter aux conditions de navigation.

L'architecture RHIZOME proposée combine les avantages des approches comportementales (e.g. rapidité de réaction face à des problèmes imprévus dans un contexte d'environnement dynamique), et ceux des approches délibératives qui tirent profit d'une connaissance *a priori* de l'environnement. Cependant, cette connaissance est uniquement exploitée pour corroborer les informations perçues visuellement avec celles embarquées. Elle est représentée par une séquence de symboles artificiels de navigation guidant le robot vers sa destination finale. Cette séquence est présentée au robot soit sous la forme d'une liste de paramètres, soit sous la forme d'un plan. Dans ce dernier cas, le robot doit extraire lui-même la séquence de symboles à suivre grâce à une chaîne de traitements d'images. Ainsi, afin de prendre la bonne décision lors de sa navigation, le robot traite l'ensemble de l'information perçue, la compare en temps réel avec l'information *a priori* apportée ou extraite, et réagit en conséquence. Lorsque certains symboles de navigation ne sont plus présents dans l'environnement de navigation, l'architecture RHIZOME construit de nouveaux lieux de référence à partir des panoramas extraits de ces lieux. Ainsi, le robot, lors de phases exploratoires, peut s'appuyer sur ces nouvelles informations pour atteindre sa destination finale, et surmonter des situations imprévues.

Nous avons mis en place notre architecture sur le robot humanoïde NAO. Les résultats expérimentaux obtenus lors d'une navigation indoor, dans des scénarios à la fois déterministes et stochastiques, montrent la faisabilité et la robustesse de cette approche unifiée.

Mots clés: Architecture de contrôle neuronale robotique, navigation autonome indoor de robots mobiles, perception visuelle, fusion de données, analyse d'un plan du bâtiment, reconnaissance de symboles, approche hybride comportementale.

Contents

1	General introduction	1
1.1	Context	1
1.2	Scope of the thesis	4
1.3	Proposed Solution	6
1.3.1	Problems addressed and contributions	6
1.3.2	Summary of the contributions	12
1.3.3	The RHIZOME architecture	14
1.4	Limitations	17
1.5	Outline of the thesis	17
I	State-of-the-art	19
2	Vision-based robot navigation	21
2.1	Introduction	21
2.2	Visual perception	24
2.2.1	Introduction	24
2.2.2	Pattern description and representation	26
2.2.3	Pattern recognition	29
2.2.4	Place recognition	32
2.2.5	Vision in robot navigation	44
2.3	World Representation	47
2.4	Localization and Path planning	49
2.4.1	Introduction	49
2.4.2	Map-based navigation	50
2.4.3	Mapless navigation	59
2.4.4	Summary of the types of navigation	63
2.5	Conclusions	64
3	Control architectures	67
3.1	Introduction	67
3.2	Control Paradigms	68
3.2.1	Design viewpoint	69
3.2.2	Functional viewpoint	70
3.3	Conclusion	87
II	The RHIZOME Architecture	89
4	RHIZOME 1: Exploring the world with little information	91
4.1	General description	91

4.2	Implementation- Rhizome 1 Architecture	94
4.2.1	Overall description	94
4.2.2	Deliberative module – Preconfigured sign sequence	97
4.2.3	Behavioral module–Neural structure	100
4.3	Experiments in real environment	122
4.3.1	Procedure	123
4.3.2	Results	123
4.3.3	Discussion	127
4.4	Conclusion	128
5	RHIZOME 2: Autonomous map-based robot navigation	131
5.1	General Description	131
5.2	Implementation- Rhizome 2 Architecture	132
5.2.1	Overall description	132
5.2.2	Deliberative module – Floor plan analysis	135
5.2.3	Behavioral module - Neural structure	145
5.3	Experiments in real environment	146
5.3.1	Procedure	147
5.3.2	Results	148
5.3.3	Discussions	156
5.4	Conclusion	156
6	RHIZOME 3: Learning and self-adapting according to unforeseen environment changes	159
6.1	General description	159
6.2	Implementation- Rhizome 3 Architecture	163
6.2.1	Overall description	163
6.2.2	Behavioral module- Neural structure	165
6.3	Unitary Experiments in real environment	182
6.3.1	Procedure	182
6.3.2	Set-up	183
6.3.3	Test and Results	185
6.3.4	Discussions	205
6.4	Functional Experiments in real environment	205
6.4.1	Procedure	205
6.4.2	Results	206
6.4.3	Discussions	213
6.5	Conclusion	214
III	General Conclusion	217
7	Conclusion	219
7.1	Summary and Contributions	219
7.2	Short-term Perspectives	224

7.3	Long-term Perspectives	226
7.3.1	Genesis of the Rhizome Architecture	226
7.3.2	Anatomy of the Rhizome Architecture	226
Publications		229
Bibliography		231
A	Appendix	257
A.1	Rhizome Architecture Optimization	257
A.1.1	Lost Sign Searching Reflex Behavior (LSSRB)	257
A.2	TOOLBOX	263
A.2.1	SIFT : Scale-Invariant Feature Transform	263
A.2.2	Bag of visual words	264
A.2.3	Adaptive Resonance Theory (ART)	267
A.2.4	Object recognition	268
B	Traduction française	271

List of Figures

1.1	Exemple of the different navigation strategies	3
1.2	Exemple of the different navigation signs found in our daily lifes.	7
1.3	RHIZOME Architecture	15
1.4	PerAc architecture(left) [Gaussier 1995]used in the behavioral module of the RHIZOME architecture (right).	16
2.1	Vision based indoor mobile robot navigation techniques proposed by [Güzel 2013]	22
2.2	Transversal structure representing the implication of different functional modules in all types of navigation strategies.	24
2.3	Visual perception. Transversal structure representing the implication of different functional modules in all types of navigation strategies.	24
2.4	Pattern Recognition process composed of image processing, followed by a pattern description and representation process and finally a pattern learning and recognition process. Image processing is not detailed in this work	25
2.5	Machine-learning techniques. Supervised and unsupervised algorithms according to traditional learning and incremental learning	30
2.6	Overall view of the place cells model proposed by [Gaussier 1997]	36
2.7	State-of-the-art local view landmark extraction. <i>Top</i> , the panoramic image taken by the robot. <i>Middle</i> , the corresponding gradient picture. Circles represent local area centered on landmark, from which small images are extracted. <i>Bottom</i> , small images after the log-polar transform,(Image extracted from [Cuperlier 2007]).	37
2.8	16 examples of 32 * 32 local views from the panoramic image (image extracted from [Gaussier 2000]).	38
2.9	Learning and recognition of the same physical landmark by several neurons. The physical landmarks M and N have been learned, for two proximal locations (the two northern crosses), as different visual patterns (upper figures). Hence, in the intermediate location (place C, lower figure), the landmarks have two valid interpretations. In location C, the activity level of «L2 and L4» for the landmark N are rather high and similar as well as the activity level of «L1 and L3» for the landmark M. (Image extracted from [Giovannangeli 2006a]). . . .	40
2.10	Azimuth of the focus points of the state-of-the-art local views. <i>Top</i> , Panorama taken by the camera. <i>Middle</i> , local views in log-polar coordinates extracted from the gradient image corresponding to the landmarks. <i>Bottom</i> , Results of the position of the local views together with the four most activated neurons with their corresponding activity value (Image extracted from [Giovannangeli 2006a]).	41
2.11	World representation. Transversal structure representing the implication of different functional modules in all types of navigation strategies.	47

2.12	Types of maps. Grid-based: collection of discretized obstacle/free-space pixels. Feature-based: collection of landmark locations and correlated uncertainty. Topological: collection of nodes and their interconnections. (Images extracted from a work presentation of [Choset 2005]).	48
2.13	Localization and Path planning. Transversal structure representing the implication of different functional modules in all types of navigation strategies. . .	50
3.1	Sense, Plan, Act (SPA) Paradigm.	71
3.2	Deliberative and Reactive Control paradigms in terms of the relationships between three primitives, sense, plan and act and in terms of how sensory data is processed and propagated through the system [Murphy 2000].	72
3.3	Hybrid control paradigms in terms of the relationships between three primitives, sense, plan and act and in terms of how sensory data is processed and propagated through the system [Murphy 2000].	73
3.4	Behavioral paradigm in terms of the relationships between three primitives, sense, plan and act and in terms of how sensory data is processed and propagated through the system [Murphy 2000].	74
3.5	Decomposition of a mobile robot control system into functional modules by following the SPA paradigm (decomposition detailed by Brooks [Brooks 1986])	75
3.6	Reactive architecture. The sensors are directly connected to the actuators . .	76
3.7	Braitenberg vehicle 1: alive. vehicle 2: afraid(2a) and aggressive (2b) vehicles [Braitenberg 1986].	77
3.8	Hybrid paradigm : most common type of architecture composed of three layers	78
3.9	Behavioral paradigm, composed of several distributed and interacting control modules called behaviors	80
3.10	Decomposition of a mobile robot control system based on task achieving behaviors (decomposition detailed by Brooks in [Brooks 1986])	81
3.11	Subsumption control architecture	82
3.12	Motor schema architecture	84
3.13	PerAc architecture by [Gaussier 1995]	85
4.1	Deduction of the directional meaning of the sign «A». The robot looks for the location of the next expected sign «B» with respect to its own position when facing the current sign	92
4.2	Functional diagram of the navigation process behavior allowed by Rhizome 1.	93
4.3	Overall view of the Rhizome 1 architecture composed of two modules: Deliberative and Behavioral	95
4.4	PerAc architecture (left) used in the behavioral module of Rhizome 1 (right)	96
4.5	Behavioral module composed of a nested PerAc module within its second level (left). As a result, the behavioral module is composed of three layers(Right)	96
4.6	Functional diagram of the navigation process behavior allowed by Rhizome 1 with the architecture layers acting on each functional decision.	98
4.7	Example of some Naomarks used in this work as the navigation signs	98

4.8	Example of some Naomarks detected in the environment by the Nao Robot [figure extracted from Aldebaran's documentation]	99
4.9	Exemple of Naomarks referenced by tag letters «A» and «B»	100
4.10	Types of links used to interconnect the neurons. The <i>one-to-one link</i> connecting each neuron in a given group to only one and unique neuron from another group; and the <i>one-to-all link</i> connecting each neuron in a given group to all neurons from another group.	101
4.11	General view of the three layers in the behavioral module	102
4.12	General view of the Sign Recognition and Movement Association (SRMA) layer	103
4.13	General view of the Direction Determination Reflex Behavior (DDBR) layer .	104
4.14	Exemple of the direction towards which, the robot needs to turn to keep the sign centered while approaching the sign. In this case towards the left since the signs is to the left side of the robot's field of view	105
4.15	General view of the Target Approaching Reflex Behavior (TARB) layer	106
4.16	Detailed view Signs Recognition and Movement Association layer (SRMA) . .	107
4.17	Activity of the sign detection group. The dynamic visual information is constantly fed into the neural group then when one or more signs appear in the robot's field of view its corresponding neuron gets activated to the maximum value.	108
4.18	Activity of the sequence sign group. The sign sequence provided by the Long Term Memory in the deliberative module is fed into the neural group. Only one neuron is activated at a time corresponding to the expected sign.	109
4.19	Activity of the Sign merged detector group. The neuron whose two input values are equal to one gets activated defined by a Heaviside function	110
4.20	Activity of a neuron u_i of the short term memory group defined by equation 4.4	110
4.21	Activity of the WTA group. Here, two WTA groups are used in order to avoid causing ambiguity when learning the association.	111
4.22	Detailed view of the Direction Determination Reflex Behavior layer (DDRB) .	112
4.23	Activity of the Learned Output Direction group. It receives the input from both WTA groups of the SRMA layer and Direction result group of the DDRB layer	114
4.24	Detailed view of the Target Approaching Reflex Behavior layer (TARB)	115
4.25	Layers convergence towards the Motor Output group.	117
4.26	Detailed view of the three layers in the behavioral module	117
4.27	Situation 1: The expected sign is detected, hence recognized, but the robot is far away from it.	119
4.28	Situation 2: The expected sign is detected, thereby recognized. The robot is close to the sign and the directional meaning is known.	120
4.29	Situation 3: The expected sign is detected, thereby recognized. The robot is close to the sign but the directional meaning is unknown	121
4.30	Situation 4: The next expected sign is detected, thereby recognized. Therefore, the movement leading the robot from sign A to B has been computed and can be associated with sign A and be learned.	122
4.31	Example of signs given in sequence by the user	123

4.32	Robot navigation within the environment	124
4.33	Summary of Rhizome 1 results obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal (RS) allowing the association learning, over time. In each of the (a, t) plots shown, (a) is the binary activation of each neural group (t) the time seconds in terms of a PerAc cycle.	125
4.34	Summary of the movements undertaken by the robot to approach a particular sign by using the TARB layer.	127
4.35	General view of the complete Architecture. Rhizome 1 is composed of two modules, one deliberative and one behavioral. This latter is composed of three reflex behaviors layers and one recognition layer	129
5.1	(a) Nao robot reading the map. (b) Map with the signs and their corresponding directional meaning. (c) View of the environment with the navigation signs in real time.	132
5.2	Functional diagram of the navigation process behavior allowed by Rhizome 2.	133
5.3	Overall Rhizome 2 architecture composed of three modules: One deliberative composed of a floor plan analysis system and two behavioral modules integrating the information from the deliberative module and allowing the robot navigation.	134
5.4	State-of-the-art floor plan analysis workflow (left) used in the deliberative module of Rhizome 2 (right).	135
5.5	Deliberative module composed of a floor plan analysis system. It consists of an information segmentation process, structural analysis process and semantic analysis process. The resulting information (sign sequence and sign directional meaning) is stored in the long term memory.	136
5.6	Two behavioral modules integrating the sign sequence information and the directional meaning each sign represents.	137
5.7	Functioning of the two behavioral modules. Since the second behavioral module integrates the sign directional meaning, the Direction Determination Reflex Behavior (DDRB) layer of the first behavioral module is no longer needed, therefore it is inhibited.	138
5.8	Functional diagram of the navigation process behavior allowed by Rhizome 2 with the architecture layers acting on each functional decision.	139
5.9	floor plan analysis process achieved through three main stages: Information segmentation, structural analysis, semantic analysis based on [Ahmed 2011].	139
5.10	Information Segmentation process.	140
5.11	Structural Analysis process.	141
5.12	Semantic Analysis process.	142
5.13	Neural implementation of a 5*5 resistive grid and the spatial memory layer (Figure extracted from [Bugmann 1995])	143
5.14	General view of the layers in the 1st and 2nd behavioral modules connected to the behavioral module.	146
5.15	General view of the Map Direction Information Reflex Behavior(MDIRB) layer.	147

5.16	Detailed view of the Map Direction Information Reflex Behavior (MDIRB) layer.	148
5.17	Floor plan of the environment. The path trajectory (green line) leading to the final destination is computed after computing a floor plan analysis process. . .	148
5.18	Set of raw images of the floor plan corresponding to different acquisition configurations of the same map.	149
5.19	Global Thresholding applied to different types of images. Before that, a black top-hat morphological operator with a 7*7 square mask is first applied on the gray-scale image. It removes noise and small object that are not relevant to the analysis in order to improve the accuracy and robustness of the image outcome.	150
5.20	Perspective correction computation. A bounding box was computed (see green square around the floor plan frame of the top images). Then, the $x-y$ coordinates of the four corners of both, bounding box and floor plan frames were calculated (colorful circles on the corners of both) and used in order to compute the perspective correction(bottom images).	151
5.21	Sign detection followed by signs and wall extraction.	151
5.22	Sign classification. The extracted signs are compared to the ensemble of nao mark signs of the dataset in order to identify them by their unique identifier ID letter.	152
5.23	left: Path trajectory leading to the final destination with the checkpoints. Middle and right: Sequence of signs computed by finding the closest sign to each one of the checkpoints.	152
5.24	Summary of the overall process chain	153
5.25	Robot navigation. While the green line refers to the path computed during floor plan analysis, the dotted purple line refers to the actual navigation path executed by the robot in real time.	154
5.26	Summary of Rhizome 2 results obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal(RS) allowing the association learning, over time. In each of the (a, t) plots shown,(a)is the binary activation of each neural group and (t) the time seconds in terms of a PerAc cycle.	155
6.1	Representation of three landmarks considered as unique within a given panoramic view. Their position is calculated with respect to a global reference.	160
6.2	Functional diagram of the navigation process behavior allowed by Rhizome 3.	161
6.3	Overall view of the Rhizome 3 architecture composed of four modules: one deliberative and three behavioral.	163
6.4	Overall view of the third behavioral module composed of a Place Cell Learning and Recognition (PCLR) layer and World Exploration Reflex Behavior (WERB) layer.	164
6.5	Functional diagram of the navigation process behavior allowed by Rhizome 3 with the architecture layers acting on each functional decision.	165
6.6	Overall state-of-the-art place cell model (blue frame) and the contributions made in our model (orange frame).	166

6.7	Neural connection between the <i>what group</i> and the <i>PrPh matrix</i> . A single neuron in the <i>what group</i> is connected to its corresponding row of neurons in the <i>PrPh matrix</i>	169
6.8	Proposed Place cell model. Composed of five neural groups. The information extracted from all natural landmarks in the panorama view are the input of the layer.	169
6.9	Overall description of the Natural landmarks Detection and Extraction process. While a batch of template images is used to build the vocabulary, the query images are used to learn and recognize the natural landmarks.	171
6.10	Exemple of Images serving to construct the vocabulary	171
6.11	Left: a query image with the ensemble of extracted key-points, each one assigned to the cluster corresponding to the nearest centroid of the vocabulary. Right: Image represented by a histogram of a total number of clusters in the vocabulary (here 100).	172
6.12	Left image: Natural landmarks resulting from spatial clusters composed of a certain number of key-points. The number of key-points varies from one another. Right image: histogram of the image according to the natural landmarks.	174
6.13	Left image: Several natural landmarks resulting from spatial clusters composed of a certain number of key-points. Each key-point has a descriptor label based on the clusters created during the vocabulary construction. Right image: histogram of landmark L1 according to its descriptors.	175
6.14	Performance of the landmark histogram expressed in the landmark vector L1 which is injected in the landmark histogram group by assigning the bin values to the corresponding neurons.	176
6.15	Activity of the landmark input neuron which value is defined by the landmark histogram group.	177
6.16	Activation of the <i>where group</i> . The <i>where</i> neuron encoding the landmark position is the one with the maximum activity value after applying a Gaussian function and its neighbours are set to a value that decays in function of their distance to it	179
6.17	Maximal place cell activation of the winner cell when the robot is at the exact same location where it has learned the place before. Exponentially decreasing activation of the others cell with respect to the distance of the robot's current location to the learned one.	180
6.18	Environment on which different tests were carried out. A certain number of positions scattered all over the test environment was chosen to code different places within it.	183
6.19	Exemple of three images out of the twelve used to create the panorama of a given place.	184
6.20	Image key-points computed by employing the SIFT algorithm.	184
6.21	Representation of the natural landmarks clustering the key-points according to their spatial proximity.	185

6.22	Representation of the images considered for the process. Since each image overlaps with the previous and next images, a filter cutting the overlapping borders was used. Therefore, only the landmarks detected within the 60 % of the center of the image were considered.	186
6.23	Computation of the position of the landmarks. It is calculated with respect to a global reference point (here an artificial landmark has been used). The computation is performed by only considering the <i>x-coordinates</i> of both landmark and reference point.	186
6.24	Test setting allowing to find the most adequate combination of SIFT parameters that would lead to a good recognition rate. The robot was positioned on place 5 where it could learn the landmark constellation surrounding it as a place. Thereafter, the robot was positioned 50 cm to the left of the learned place and the recognition activity was computed.	188
6.25	Recognition rate % of place 5 (a_{p_5}) when the robot is placed 50 cm away from it. The values of the SIFT parameters (contrast threshold, edge threshold and sigma) allowing a good description of images are modified one at a time. . . .	189
6.26	Computation of the recognition rate % of place 5 (a_{p_5}) by combining the parameters values that gave the highest recognition rates in the first set of results.	189
6.27	Computation of the recognition rate % of place 5 (a_{p_5}) by fixing the sigma parameter to the two values (9.6 and 4.8) giving the highest recognition rate in the first series of tests (6.25 while testing with different contrast threshold values.	190
6.28	Computation of the recognition rate % of place 5 (a_{p_5}) by fixing the contrast threshold parameter to the three values giving the highest recognition rates in the previous tests(0.07, 0.14 and 0.16) while testing with different sigma values.191	191
6.29	Comparison of the recognition rate of each set of contrast threshold-sigma pair by setting the value of the edge threshold parameter to each of the two values that gave the best results in the previously presented individual test. .	191
6.30	Representation of the environment test illustrating the learned place and the surrounding spots where the robot was placed to test and compute the recognition rate in % of place «5» corresponding to $a_{p(0,0)}$	192
6.31	Results of the recognition rate in % of place $a_{p(0,0)}$ when placing the robot at different test places (by following the same topology given by the environment test) computed with three different values of sigma-contrast threshold pair. The values which have been highlighted are those with the maximum recognition rate at each place among the three test.	193
6.32	Test environment illustrating the different places at which the robot was placed. {2, 4, 5, 6 and 8}	194
6.33	First set of bach learning tests. The robot learned the place 2. Then for each of the different places {2, 4, 5, 6 and 8} its recognition rate was computed a_{p_2} . 194	194
6.34	Second set of bach learning tests. The robot learned the place 4. Then for each of the different places {2, 4, 5, 6 and 8} its recognition rate was computed a_{p_4}	195

6.35	Third set of bach learning tests. The robot learned the place 5. Then for each of the different places {2, 4, 5, 6 and 8} its recognition rate was computed a_{p_5} .	195
6.36	Fourth set of bach learning tests. The robot learned the place 6. Then for each of the different places {2, 4, 5, 6 and 8} its recognition rate was computed $a_{p(6)}$.	196
6.37	Fifth set of bach learning tests. The robot learned the place 8. Then for each of the different places {2, 4, 5, 6 and 8} its recognition rate was computed $a_{p(8)}$.	196
6.38	Tests performing learning all places in the environment {2, 4, 5, 6 and 8} and computing their recognition rate by placing the robot at each of the learned places at a time.	197
6.39	Recognition activity of all places when the robot was positioned at place 4 (see TEST #6.2.).	198
6.40	Three different places at which the robot was placed. It learned each of them in an incremental way.	199
6.41	Recognition rate results of the learning sequence as the robot walk along the line covering places 4^{th} , 5^{th} and 6^{th} over the time.	199
6.42	Activity of the three learned places over time.	200
6.43	Recognition rate results of the learning sequence as the robot walks along the line covering places 2^{nd} , 5^{th} , 4^{th} , 8^{th} and 6^{th} over time.	201
6.44	Test environment with the learned places designed in red circles and the tested places in orange squares. The red trace represents the path navigated by the robot when learning the places in the corresponding order: {2, 5, 4, 8, 6 }. The results table,designed to the right, follows the same topology given by the environment test. The learned places in pink and the places where the robot was positionned to test the recognition rates in blue.	202
6.45	Recognition rate of all places learned incrementally and in a batch way when the robot is positioned at places 11^{th} and 12^{th}	202
6.46	Recognition rate of all places learned incrementally and in a batch way when the robot is positioned at places 10^{th} and 13^{th}	203
6.47	Recogniton rates of different places learned in different order. Only three places were learned in both ways (batch and incremental)but in different order every time and the recognition rate of all of them was tested while positioning the robot at place 13^{th}	204
6.48	Test navigation environment. Only two signs from the sign sequence are placed in the environment.	206
6.49	Summary of Rhizome 3 results obtained when the signs and directions are extracted from the floor plan. They are obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal (RS) allowing the association learning, over time. In each of the (a, t) plots shown,(a) is the binary activation of each neural group and t the (t) time seconds in terms of a PerAc cycle.	209

6.50	Summary of Rhizome 3 results obtained when only the signs are given by a command program. They are obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal (RS) allowing the association learning, over time. In each of the (a, t) plots shown, (a) is the binary activation of each neural group and t the (t) time seconds in terms of a PerAc cycle.	212
7.1	Rhizome 1 composed of a basic deliberative module and a behavioral module composed itself of three layers.	221
7.2	Rhizome 2 composed of a deliberative module and two behavioral modules. The deliberative module is in charge of computing a document analysis process on a floor plan. The first behavioral module correspond to the one presented in Rhizome 1 and the second is in charge of process the directional meaning each sign denotes extracted from the floor plan.	222
7.3	Rhizome 3 composed of a deliberative module and three behavioral modules. The deliberative module stores the navigation sign sequence that is either given by a command program or extracted from a floor plan. The Behavioral modules use both <i>a priori</i> and dynamic visual information allowing the robot to navigate based on the recognition or absence of the artificial navigation signs. In their absence a place recognition system is performed.	223
7.4	Representation of the imbricated set-up of the RHIZOME architecture. While Rhizome 1 can be considered as a square structure, Rhizome 2 composed of this same square, forms a cube and Rhizome 3 a hypercube that encompasses Rhizome 2 which itself encompasses Rhizome 1.	224
A.1	Detailed view of the Lost Sign Searching Reflex Behavior (LSSRB) layer composed of seven neural groups.	258
A.2	Activity and connection of the Sign Lost Memory group connected to the Sign sequence group of the SRMA layer and the Sign Lost detector group connected to the first WTA group of the SRMA layer.	259
A.3	Activity of the trigger search group. When activated, it inhibits the activity of the reflex movement performed by the DDRB.	259
A.4	Activity of the Memory Position group and its connection to the output units. The position is stored according to the last opposed movement. For instance if the robot last movement was to the right, then the movement to be performed to find again the sign would be to the left and Vice-versa	260
A.5	Convergence of both reflexoutput Search groups (left and right) in the Motor Output (MO) group. A Reflex movement to the right neuron has been added to the final Motor Output(MO)	261
A.6	Connectivity of the three reset neuron groups used in the overall architecture. Here, the third reset Memory has been added	262
A.7	General view of the complete Architecture. Rhizome 1 is composed of two modules, one deliberative and one behavioral. This latter is composed of three reflex behaviors layers and one recognition layer	262

A.8	SIFT descriptor construction with its corresponding histogram.	265
A.9	Bag-of-visual words. First, the construction of a vocabulary allowing to identify the visual words by clustering the whole set of features is extracted (1.a) according to their similarity (1.b). Second, the assignment of features of the new images to the cluster with the closest centroid (2.a), followed by the histogram of the number of occurrences of the features in the given image (2.b).	266
A.10	ART Adaptive Resonance Theory	268
A.11	Object recognition litterature divided into windows-based and part-based object representation	270

General introduction

Contents

1.1	Context	1
1.2	Scope of the thesis	4
1.3	Proposed Solution	6
1.3.1	Problems addressed and contributions	6
1.3.2	Summary of the contributions	12
1.3.3	The RHIZOME architecture	14
1.4	Limitations	17
1.5	Outline of the thesis	17

1.1 Context

A large extent of research in the last four decades has been focused on the conception of robust, flexible and reliable autonomous robots capable of traveling from a starting point to a goal. Robot navigation can be defined as the process allowing a mobile robot to move autonomously from a starting point towards a final destination by using sensorial data in an environment that, in most of the cases, is dynamic and unpredictable.

Mobile robots such as wheeled and bipedal robots are widely being used for a variety of applications in military, industry, service and domestic environments. There exist different navigation approaches of achieving mobility in robots, but they all serve the common purpose of leading the robot to its final destination in a safe way.

In order to better understand these approaches, as well as the motivation behind this work, the requirements of navigation are presented using the analogy to a daily life navigation scenario below(see figure 1.1).

Imagine that you arrive for the first time to a large unknown building and you are required to find your way within it to get to a final destination. Depending on the purpose of your visit, the time constraints and the availability of the information about the building, different strategies can be considered.

Firstly, you can use a map of the building (***map-based navigation***). With it, you can plan in advance the path trajectory leading you to the desired destination (*off-line planning*). You can either read the map that is at the entrance of the building, get it printed in a paper-based form, or, in a rare case, have it on a phone or tablet in a digital form. In the first case, as you can only see it once, it would require you to first memorize the distance to walk as well as the landmarks or places you might encounter so that they can be looked in the environment while navigating. This case is similar to as if someone else, already knowing the place, would have given you some indications (e.g. landmarks and distances) to get there. In the two latter cases, the memorization task will not be necessary as you can just match the map information (usually landmarks) continuously with what you perceive while navigating.

Alternatively, you can build your own map (***map-building navigation***), as you navigate the building and learn the path trajectory (*on-line path planning*) leading you to the final destination. In either case (***map-based*** or ***map-building navigation***), maps can contain either all dimensions of the environment including distances between places within the building (*metric map*) or just an arrangement of the important landmarks and places according to their proximity (*topological map*). There is also the option to find the destination without the support of a map (***mapless navigation***) by exploring the building at your ease. By memorizing the different landmarks or places seen while walking, you could build your own path trajectory as you go further (*on-line path planning*) until finally arrive to your destination.

Understandably, any of the above approaches can be used in both indoors and outdoors environments, but they all have their pros and cons.

For instance, having a map of the building before the navigation activity starts would allow you to reach the desired destination fast, while knowing at all times your position with respect to both starting and final points. However, it requires that somebody has previously drawn the map of the building, and it has remained the same since it was drawn.

Instead, building your own map based on what you are currently perceiving would provide you up-to-date knowledge about the building as well as a certain autonomy compared to the first approach. Thus, in case of any recent change of the environment you could adapt to it and store the path for the next time the information is needed. However, building your own map would take you a lot of time and effort especially if your own position is not clearly known with respect to the starting or the final point.

Similarly, navigating without any map would allow you to find new and different ways of accessing the desired destination in case that one of them has been blocked for any reason. However, this task would require you to have sufficient time to explore all possible ways. Additionally, since there is no recording of the environment, returning to the starting point or coming back another day would be as time consuming as it is the first time. Moreover, if

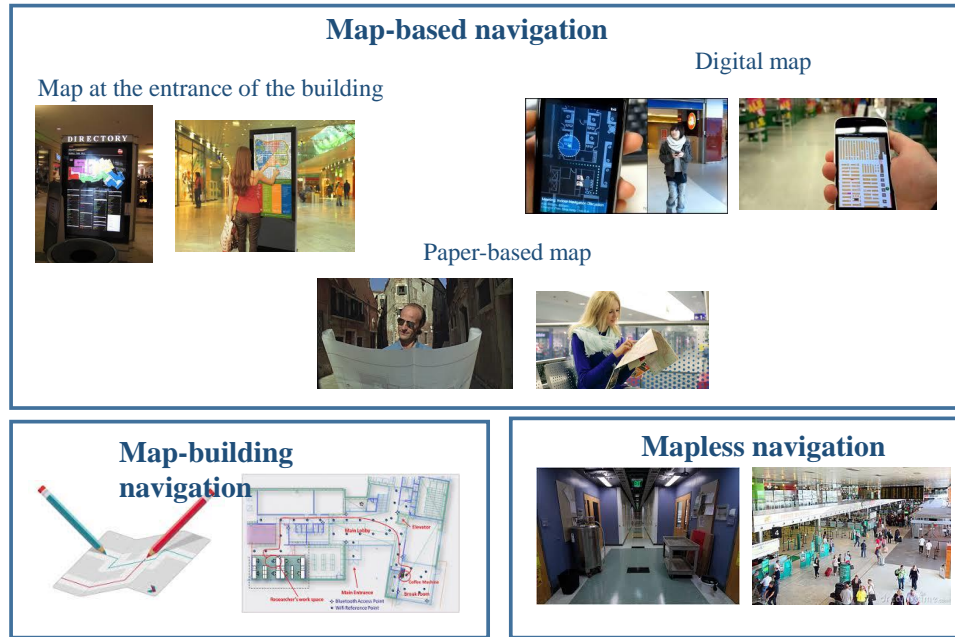


Figure 1.1: Exemple of the different navigation strategies

you get lost on the way, it would be difficult to know where your location is with respect to the starting point and the final destination.

In the context of autonomous robot navigation, many systems attempt to give a solution to the navigation problem by employing any of the above navigation strategies. The navigation process usually involves the use of **world representation**, **localization**, **path planning**, in order to **execute** the appropriate action according to the **perceived** environment information without continuous human guidance.

Each functional module is by itself a vast research field and the existing relationships among them and means of integration are defined by a control architecture. For instance, one or some of them can be omitted in the overall process depending on the control architecture or the navigation strategy employed. For a general overview or detailed description of these modules, the reader can refer to the appendix section and chapter 2 respectively.

Control architectures are the core of successful navigation as they impose constraints on how the system can be controlled. There are considerably many possible ways to program a robot and currently, not a single control architecture can be said to excel at performing in all possible applications.

The complexity of control architectures is subject, to a large extent, to the special needs of robot systems that most of the time must be fulfilled concurrently and asynchronously. How they succeed at organizing, unifying and monitoring the aforementioned modules to enable an effective navigation strategy is highly determined and limited by both the application field and the robotic platform implementation. Consequently, robotic systems can be divided according to both application field and robotic platform.

From the application field point of view, different scenarios to which robot systems (e.g. industrial robots, domestic or household robots, service robots, military robots, space robots) are exposed need to be considered. For instance, in deterministic scenarios where the environment remains intact, the navigation task can be simplified to the act of just going from a starting point to the final destination by sensing the environment and/or following the planned path if a representation of the world is previously provided. On the contrary, in stochastic and thereby more realistic scenarios such as those found in search and rescue applications, robots systems must cope with unexpected situations by reacting accordingly and responding within varying temporal scopes when performing real time navigation. They must have decision-making capabilities to act in accordance with multiple and different tasks while still achieving to the final destination.

From the robotic platform implementation point of view, robot systems (e.g. wheeled robots, bipedal (humanoids) robots, marine (underwater) robots, aerial robots), need to control diverse physical components such as sensors, actuators, processors. in order to interact with uncertain and often dynamic environments in real time. Additionally, the quick growth of the embedded power computing requires the architectures to be flexible enough to allow the replacement of components in the field when necessary and easily adaptable to different platforms in order to perform different missions while still maintaining an autonomy.

All these facts must be taken into account as their impact is substantial for the conception and development of an autonomous robot navigation system. Hence, in order to manage such complexity, many architectures have been proposed, along with software and material components needed to support them.

Most control architectures have shown to excel at performing a successful autonomous navigation in a specific application field despite the constraints of the platform employed. However, they usually fail when a different scenario is presented and the platform requirements are no longer adequate for the given task.

1.2 Scope of the thesis

Since a multitude of situations can lead to a diversity of architectural solutions and related mechanisms, an ideal solution would be to conceive and implement a control architecture as generic as possible, which can overcome all different sort of constraints. Thus, such architecture should be flexible enough to allow the addition of new components (hardware and software) without questioning or modifying the already existing ones at whatever level they might be; and consequently offer transparent mechanisms of communication and exchange of data. It should also be capable of a high-level decision-making capacity to perform the adequate actions while being able to refine and adapt its plans and its behaviors according to its goals and to the unpredictable environment changes. Moreover, since such architecture would be capable of performing (sometimes concurrently) multiple actions, it should also be robust enough to manage the priorities of different tasks, the sudden malfunctions and the redundancy of the information given by the multiple sensor sources while still guarantying a safe performance.

The list of these properties is endless and after a certain point, it is impossible to make

it exhaustive since our capacity of imagining all kind of situations to build an all-purpose autonomous robot system is quite limited by many reasons (i.e. technological, societal, ethical, etc). We could conversely, think of an architecture that is not only endowed of the established properties permitting to respond to different constraints but also and overall, an architecture capable of transforming itself according to any new unforeseeable parameter.

The author of this work is aware of the fact that succeeding in conceiving and implementing such generic architecture is not an easy task and that it will require a lot of time, effort and resources to get there. Therefore, the work presented in this thesis must be regarded as a step on the path towards finding a more general, highly robust and efficient approach to control robot architectures that allows autonomous robot navigation in a variety of scenarios. To this end, this work focus exclusively on the constraints given by the different application fields.

To summarize, the main goal of this research thesis is to propose a new robotic control architecture capable of easy adaptation to different scenarios where a robot is able to navigate towards its final destination while coping with possible unexpected situations. To this end, the work presented in this thesis has been conducted within the context of an indoor visual-based robot navigation with the application on a humanoid robot using a camera as the only sensor to perceive the environment. Two completely different scenarios (from which many others can derive) are here considered:

1. A **deterministic** scenario supposing that the environment remains always the same,
2. A **stochastic** scenario presenting unforeseen changes in the environment.

The distinction of these two scenarios suggests that they both are defined vis-à-vis a prior knowledge of the environment and the possibility of matching or not with it. Hence, during navigation, the robot looks for the expected information in the navigation path; and by performing the respective matching with what it is currently being perceived, it can corroborate the information analogous to the map-based navigation strategy.

1. Thus, when the matching along the way to the final destination results positive, it can be deduced that the robot is navigating within a **deterministic** scenario.
2. Otherwise, the scenario is assumed to be **stochastic** and the robot is obliged in return to find a way to get to its final destination by following some mapless navigation techniques.

From its creation through its development and completion, this work has its foundation on two important pillars: the state-of-the-art **indoor navigation strategies** based on visual perception and the different paradigms that **control architectures** are based on.

Indoor navigation can be divided into *map-based*, *map-building* and *mapless navigation*. While *map-based navigation* provides a model of the environment before the navigation starts, *map-building* builds one as the robot navigates the environment. *Mapless navigation* systems instead, do not employ any representation of the world and the movements of the robot depend on the elements observed in the environment.

The paradigms of **control architectures** can be classified according to two different viewpoints: functional and design viewpoints. The functional viewpoint classifies the paradigms in terms of their internal functionality and thus in terms of their capabilities to act on the environment when performing a given task. These are *deliberative*, *reactive*, *hybrid* and *behavior-based*. The design viewpoint specifies two paradigms based on how the data information is processed and propagated through the systems as well as how the knowledge is ordered: *top-down* and *bottom-up*.

For a deeper insight, the reader can refer to the state-of-the-art presented in Chapter 2 and Chapter 3 for vision-based robot navigation and control architectures respectively.

1.3 Proposed Solution

In order to meet the above objective according to the specificities given by the different scenarios, the RHIZOME (**R**obotic **H**ybrid **I**ndoor-**Z**one **O**perational **M**odul**E**) control architecture is here proposed.

The architecture is composed of an artificial neural network comprised of interconnected artificial nodes that compute their output values from one or more inputs received, akin to a biological neural network in the nervous system of an organism. These computational models have properties such as associative memorization, learning and parallel multi-information processing. Their interconnectivity is such that the whole system is capable of self-adapting with regard to its inputs resulting from the interaction between the robot and the environment.

During the conception and implementation of the RHIZOME architecture, a number of questions, knowledge gaps and decisions points needed addressing in order to choose the best solutions among the existing ones. This is given next, via sequential addressing of some of the most relevant questions, before getting into the details, characteristics and functioning of the RHIZOME architecture.

1.3.1 Problems addressed and contributions

Conceiving and implementing a control architecture capable of adapting to different scenarios requires satisfying various very different requirements. For that purpose, the problems addressed in this thesis were reasoned according to the aforementioned working scenarios (deterministic and stochastic) constraints. A logical approach was used to flow down from one question or decision point to the next, gradually firming up the solution space to the problem at hand, in the form of the RHIZOME architecture. This logic is presented below to justify the decisions undertaken by providing the rationale behind them.

1. **How can the world (navigation environment) be used in the best way possible to help a robot to navigate?**

Going back to the daily life analogy described above where three different navigation strategies can be considered to get to the final destination, it should be noted that there is a common denominator: the use of salient and most of the time stationary cues serving as reference points. Certainly, when using a map, the information (after having planned the complete path) can be just reduced to the distance to walk and the different directions to take. However since most environments are prone to unforeseen changes, perceiving the environment is necessary to corroborate that the information given by the map has not been modified. Likewise, when no representation whatsoever of the world is provided in advance, those cues play an important role in the navigation task.

This can be seen from the early days of sea navigation where reference points were essential to sailors to localize themselves and navigate in the oceans. Before the arrival of modern navigation techniques, the pole stars were used as reference points for navigating since they did not disappear in the horizon. Similarly, in our daily life, cues (referred as navigation signs hereafter) are found all around us to help with our navigation chores. From traffic signs on the roads, directional signs in metro stations and airports to fire safety signs in the buildings, all these signs help in a way or other to guide us through the destinations we want to achieve (1.2).



Figure 1.2: Example of the different navigation signs found in our daily lives.

Based on this insight, this work makes use of some **navigation signs** for reference purposes in the navigation task of a mobile robot. By detecting and recognizing such signs while navigating the environment, the robot is guided through to achieve its final destination. Two types of navigation signs are here considered: **Artificial signs and natural signs**. Artificial navigation signs refer to predesigned signs previously placed

along the navigation path. Conversely, natural navigation signs refer to natural patterns within the environment sufficiently remarkable to be considered as reference points.

The use of both types of navigations signs can be considered from the point of view of two completely different paradigms that have their roots in the field of artificial intelligence. First, the symbolic paradigm of artificial intelligence assumed that the manipulation of symbols was sufficient to treat many aspects of intelligence such as the control of complex actions in the machines. This can be true and has proven to be robust over the years when the output of a machine manipulating symbols are based on definite inputs and when there is certainty (e.g. expert systems, deliberative robotic control architectures, etc.).

This was a predominant paradigm over almost three decades from the mid-1950s. However, their lack of robustness under uncertainty led new researches to look into a sub-symbolic approach: the *nouvelle AI* paradigm. Contrary to the symbolic paradigm, the *nouvelle AI* paradigm does not use a specific representation of the world to approach intelligence, but instead it postulates that intelligence emerges from simple behaviors resulting from the interaction of the robot with its immediate environment via sensory-motor links. Rather than using a single centralized planner, the system stores representations in a distributed fashion over multiple behaviors, while the overall ensemble composes an interconnected behavior network. Hence, each behavior, representing a component of the whole representation, communicates with other representation behaviors as well as utilizes other low-level behaviors.

Hence, while the artificial navigation signs are used in this work as high-level symbols that the robot can manipulate under a deterministic scenario, the natural navigation signs resulting from the interaction of the robot with the environment are used in the absence of the artificial navigation signs under a stochastic scenario. This configuration result from the following questions and their converging answers.

2. How can the robot access its final destination in the most efficient and simple way?

The simplest scenario that can be imagined is one in which the representation of the world is computed beforehand and it is found as such during the navigation task. Such configuration implies that the scenario is deterministic and the representation of the world is provided to the robot before the navigation activity starts.

From the navigation point of view, a **map-based navigation** strategy is suitable for this case since it uses a representation of the world in advance.

From the architectural point of view, a **deliberative architecture** and **top-down processing** seems to be more suited to process the information given by the map and plan the navigation path. In effect, deliberative architectures were the dominant paradigm for building robots based on a previously given model and process the information in a top-down fashion. In our case, the model corresponds to the environment within which the robot needs to navigate.

- **But what type of map?**

Digital floor plans of buildings (such as hospitals, schools, residential complexes or factories) are typically not as readily available as digital maps of entire cities, countries etc. Hence, when trying to navigate an unknown building, one has to rely on floor plans available in physical form at the entrance of the building or on paper to achieve one's final destination. Since such a **floor plan** provides one of the fastest way to access comprehensive information about the inside of the building, it is the type used here to represent the world information (in this case, artificial navigation signs designed on the map according to their placement in the navigation path). Then, computing a **sequence of these signs** according to their order of appearance within the path from the starting point to the final destination and then looking for them in the same order along the navigation path seems to us to be most suitable option to obtain the information. This implies a **topological** configuration of the map, where the only information needed is the sequential relationship among the signs.

In the case the sequence of signs is the only thing provided to the robot, there are some chances to fall into the «symbol grounding problem» [Harnad 1990]. This problem refers to concerns and issues such as: What is the directional meaning of each sign? Would it be the same if they were found in another environment? How can the robot interpret and deduce their meaning? All these questions or problems can only be answered by allowing a constant interaction between the robot and the environment

Conversely, if the robot has the autonomy of "reading" the map by itself, it could compute not only the sign sequence but also the directional meaning each sign denotes as well as the distance among the signs. To this end, some metrics usually found in **metric** maps would also be needed.

Regardless of the information obtained (signs sequence only or signs sequence with their corresponding directional meaning), the interaction with the environment becomes essential in this task as the robot needs to compare the obtained information to what it perceives in the real environment. The robot should not only be able to corroborate the information given by the map (sign recognition) but also, it should know what action to perform among different and multiple other possible actions resulting from recognizing (or inability to recognize) the sign.

Interacting with the environment is quite challenging, as it is most of the times dynamic and unpredictable; and even though the use of *a priori* information might alleviate the navigation task, a good system should be flexible enough to cope with any potential unforeseen change in the environment. Therefore, it is necessary to find out the best solution allowing handling the uncertainty problem and thereby the execution of a given action among many other. This takes us to the next question.

3. What type of mechanism or model seems to be well suited to handle dynamic environments?

In order to tackle the uncertainty problem given by dynamic environments, the best model one can think of is that of the human brain. In effect, humans have shown an extreme capability to handle unforeseen changes by reacting accordingly, learning by example and from experience and easily adapting to any given situation. Such successful performance can be attributed to the thorough work of the brain and its components. Hence, the properties, behavior and functionality of the neurons composing the brain have been the inspiration of several models in artificial intelligence and they represent the foundation of the internal components of the proposed architecture.

The neural models of the connectionism formerly known as the Parallel Distributed Processing or PDP models [McClelland 1986], use simple and often uniform neuron-like processing units to process the information. The memory is carried locally through the interaction of a large number of these units via excitatory and inhibitory signals. Each unit receives input from its neighbors, executes a function according to the received inputs and computes an output value. The inherently distributed configuration allows the computation of several units to be carried out simultaneously which compared to serial models, allows to hasten the information processing.

The representation of the knowledge in PDP models is not stored in a state or a long-term memory as it can be found in other conventional models. Conversely, the knowledge is part of the process itself and determines the course of it in the sense that it is stored in the connections strengths among units as a long-term memory, while the short-term memory is stored in the states of the units. The units may represent different things depending on the model. For instance, a simple unit can represent a feature, a symbol or a concept. It can also represent abstract elements, which by assembling it with many other can represent an entire feature or concept.

An extremely important property of these models is that it is possible to learn through experience by using a modulation mechanism allowing to adjust the connection among the units. There exist different rules for adjusting the connections. Most of them derived from the learning rule proposed by [Hebb 2005] who stipulated that when two units are simultaneously excited the connection between them is strengthened.

From the architectural point of view, behavior-based architectures are composed of a collection of behavioral modules organized in a distributive and parallel fashion alike the PDP models. They are usually executed concurrently and asynchronously and by bringing them together under complex environments, emergent behaviors can occur. Since there is no central control among the behavioral modules, all layers are interconnected allowing an internal communication to decide on the best action or behavior to be performed [Edelman 1987].

Therefore, a **behavior-based architecture** with a **bottom-up processing** seems to be a good option to tackle the uncertainty of real-time interactions under dynamic and unpredictable environments, in particular given the great achievement of the subsomption architecture of Brooks [Brooks 1986] at overcoming such problems.

This work follows the same line of thought of the behavior-based architecture in order to adapt to new changes and act accordingly. More particularly, the mechanism, prop-

erties and components of the proposed architecture are based on the PerAc architecture Inspired mainly by the works of Brooks [Brooks 1986], [Edelman 1987],[Carpenter 1987] and proposed by Gaussier and Zheren [Gaussier 1995] as an organized neural structure.

The PerAc architecture makes no use of any representation of the world to control the action of the robot. Conversely, it follows a perception-action mechanism that constantly evolves because of the dynamic interaction between the robot and its environment (see section 3.2.2.4 in chapter 3 for more information).

- **What action can be performed in case the expected information is not seen in the environment?**

The absence of the artificial navigation signs within a stochastic environment forces the robot to opt for a new navigation strategy requiring it to find new reference points to follow its way. To this end, finding natural navigation signs as a result of the interaction of the robot with the environment as found in **mapless navigation** strategies seems to be a good solution.

However, this work goes beyond the detection of natural navigation signs by using a more robust system based on a biologically inspired approach proposed by [Gaussier 2002], which allows place recognition. In effect, a place can be identified as a stable reference point that can be learned by keeping in memory the location of the most relevant perceived patterns within the panoramic visual field of the robot. Returning to this place then consists in navigating until recognizing the same learned patterns.

The robustness of such approach lies in the fact that even if one or several patterns characterizing the place are removed or not visibly available anymore, a place can still be recognized. Additionally, by means of a triangulation process it is possible to obtain information about the robot's position with respect to the surrounding environment.

Finally, the fact of knowing the action to perform when the expected information is not seen in the environment implies that there has been a process allowing to compare both source of information (the *a priori* and the perceived real-time information while navigating) and that there has been a choice of executing an action according to the result. Thus, a final question arises as follows.

4. **How to merge both input information and use it to act according to the resulting comparison?**

The solution to this question may be seen as the convergence of all the above questions and answers into a single and unique structure. How to combine *a priori* and real-time dynamic information? how to combine both deliberative and behavior-based architectures? Alternatively, how to combine the use of different techniques from mapless and map-based strategies?

All such questions can be covered by answering to a more general question: how can a connectionist model be combined with a symbolic computation model?

Over the years, both views have been considered opposite to each other. Whereas the knowledge information is stored in the connection strengths among the network units of the connectionist approach, the same knowledge is represented by strings of symbols in the classical symbolic approach.

Despite the differences, some connectionists [Sun 2001a],[Sun 2001b] agree that it is possible to reunite both paradigms into a connectionist architecture. They postulate that it should be possible to implement a symbolic processing in a neural network given the ability of humans to perform high-level symbol-manipulations tasks despite the neural net configuration of the brain.

Following the same line of thought, the architecture proposed in this work takes advantage of the properties characterizing the neural networks to merge both information into a neural structure.

Three types of units have been distinguished in classical neural network models: input, output, and hidden units. In this work, the author has opted for naming internal units what it could seemingly be the hidden units, in order to avoid any association to the configuration of the most commonly known models of neural networks such as the recurrent Neural networks (RNN) or the Feed-forward Neural Networks. Indeed, contrary to those models, the presented architecture takes the liberty of connecting the units and group of units in a distributed fashion different from what one can be used to see. The reader can refer to chapter 4, which explains in detail the connections of such units within the context of our work.

As far as the action-selection problem is concerned, the properties of the artificial neurons give an inherent solution to it. In his book, the mindful Brain [Edelman 1987], Edelman develops his theory of neural Darwinism, where he evokes the plasticity in the neural networks in response to the environment. The interconnection among the neurons is reinforced through experience and when a external or internal stimulus is received by the system, different neurons are simultaneously activated sending the information to their neighbors. The output is then the result of the different activations stimulated by a given input source.

Artificial neural network models have the ability to model any given function. Therefore, it is possible to set different activation functions along the network in order to trigger different behaviors.

As a result, the whole system works in parallel and a «competitive mechanism» allows deciding on the best behavior or action to perform for controlling the robot according to the stimulus received [Carpenter 1987],[Kohonen 1990].

1.3.2 Summary of the contributions

The fact of analyzing, reorganizing and synthesizing solution areas as presented above, led the author to understand that the conception of a unique control architecture capable of responding to different scenarios constraints is only possible by conciliating all differences among the so-far-proposed paradigms. Thus, rather than embracing a single approach or

following a single path of thought, one can think of creating a synergy of multiples approaches by merging them into a transversal structure.

The whole set of contributions of this work is next summarized. They follow the different approaches that served to achieve the final goal of representing and conceiving such hybrid configuration .

Navigation Viewpoint: From the navigation point of view, two navigations strategies are used.

Map based: The analysis of the floor plan in real time undertakes a thorough process permitting the robot to extract the relevant information for its integration into the system. It consists of (1) an information segmentation process, which identifies and separates different types of information; (2) followed by structural analysis where the information is extracted (walls and navigation signs separately); (3) and finally a semantic analysis allowing the extraction of the sign sequence based on the computation of the path and the information of the signs. While the floor plan is designed in a topologic fashion, the extracted information is both topologic and metric.

Mapless: A biological approach for place recognition based on place cells is implemented. Firstly, the procedure for detecting the landmarks undertakes two-classification process. The SIFT local descriptor [Lowe 2004] and a visual bag of words model are first used in order to describe distinctly the salient features of all the images. Then the features are clustered according to their proximity in terms of distance and the resulting group is considered as the salient landmark. Finally, each landmark is compared to others by computing the norm of the difference between the features describing them. Secondly, the internal computation of the neural components are modified in order to allow the robot to compare the landmarks perceived from different places during navigation, by using a vigilance term inspired by the work of Grossberg [Carpenter 1987] and learn them when not recognized. Consequently, the system learns incrementally.

The third navigation strategy, *Map building*, is only presented as a perspective for future work. At the end of the navigation, the robot is capable of updating the map with the new information given by the place cells. The map is built by merging both static map information and recently-changed information. The methodology used correspond to the map-building training phase. A SLAM technique could also be foreseen.

Architectural Viewpoint: Two points of view can be considered from which all types are in a way or other used.

Functional Viewpoint: Contrary to the PerAc architecture, the RHIZOME architecture uses an *a priori* knowledge of the environment in order to corroborate the dynamic visual information perceived during navigation. Hence, it is composed of both **deliberative** and **behavior-based modules** interconnected by a neural network which makes of it a **hybrid** architecture. However, the hybrid sense here opposes to the currently known hybrid architectures that use an intermediate component to reconcile both representations and to resolve any conflict between their outputs. It acts as the coordinator of the system and it plays an important role in the good performance of the system.

Conversely, the RHIZOME architecture can be considered as being entirely behavior-

based capable of combining two opposing approaches without the need of a coordinator component. Hence, a behavior-based hybrid architecture.

However, it differs from the common behavior-based control architectures in the fact that this architecture does not follow a hierarchical process but instead, each action or behavior is equally important and the resulting action emerges from the interaction with the environment and the internal motivation of the robot.

Design Viewpoint: Whereas the information available from the map is obtained by following a **top-down process**, the emergence of behaviors and actions of the robot result from a **bottom-up process**.

Consequently, such hybrid or multi-hybrid configuration, if one might say, results in the conception of a complete architecture imbricating different architectures each suited for a different scenario.

1.3.3 The RHIZOME architecture

The RHIZOME architecture emerged out of the will to provide an adequate autonomy to mobile robots allowing them to navigate within an environment while being capable of adapting themselves to unforeseen situations presented in it. It consists of a behavior-based hybrid architecture that fuses the *a priori* information and real-time visual information of the world into a neural structure.

The *a priori* information of the world is used to only corroborate the real-time visual information perceived during navigation, contrary to most hybrid architectures that use it to directly control the actions of the robot. Additionally, instead of using a complete motion path, the RHIZOME architecture makes use of artificial navigation signs and their expected sequence in the navigation path. Consequently, in order to take the right decision during navigation, the robot is able to process both set of information, compare them in real time and react accordingly. When the navigation signs are not present in the navigation environment as expected, the RHIZOME architecture allows the robot to learn and recognize places based on natural navigation signs that it perceives in the environment. Thus, the robot is still able to achieve its final destination by overcoming the unforeseen situations. The RHIZOME architecture is composed of a hybrid behavioral structure that combines a deliberative module and one or several behavioral modules as illustrated in figure 1.3.

- On one hand, the **deliberative module** represented by the top half box of figure 1.3 corresponds to the *a priori* knowledge of the navigation environment. In this work, it is given in the form of navigation signs that are expected to be found in the navigation path. The sign sequence is computed beforehand according to the order of appearance of the signs within the path from the starting point to the final destination. It is either integrated into the behavioral modules of the architecture through a command program or by following a processing chain in charge of extracting it by computing a path plan from a given map (floor plan).
- On the other hand, the **behavioral modules** represented by the lower box of figure 1.3 are based on the PerAc (Perception-Action) architecture [Gaussier 1995] which is composed of two levels of data streams corresponding to perception and action flows.

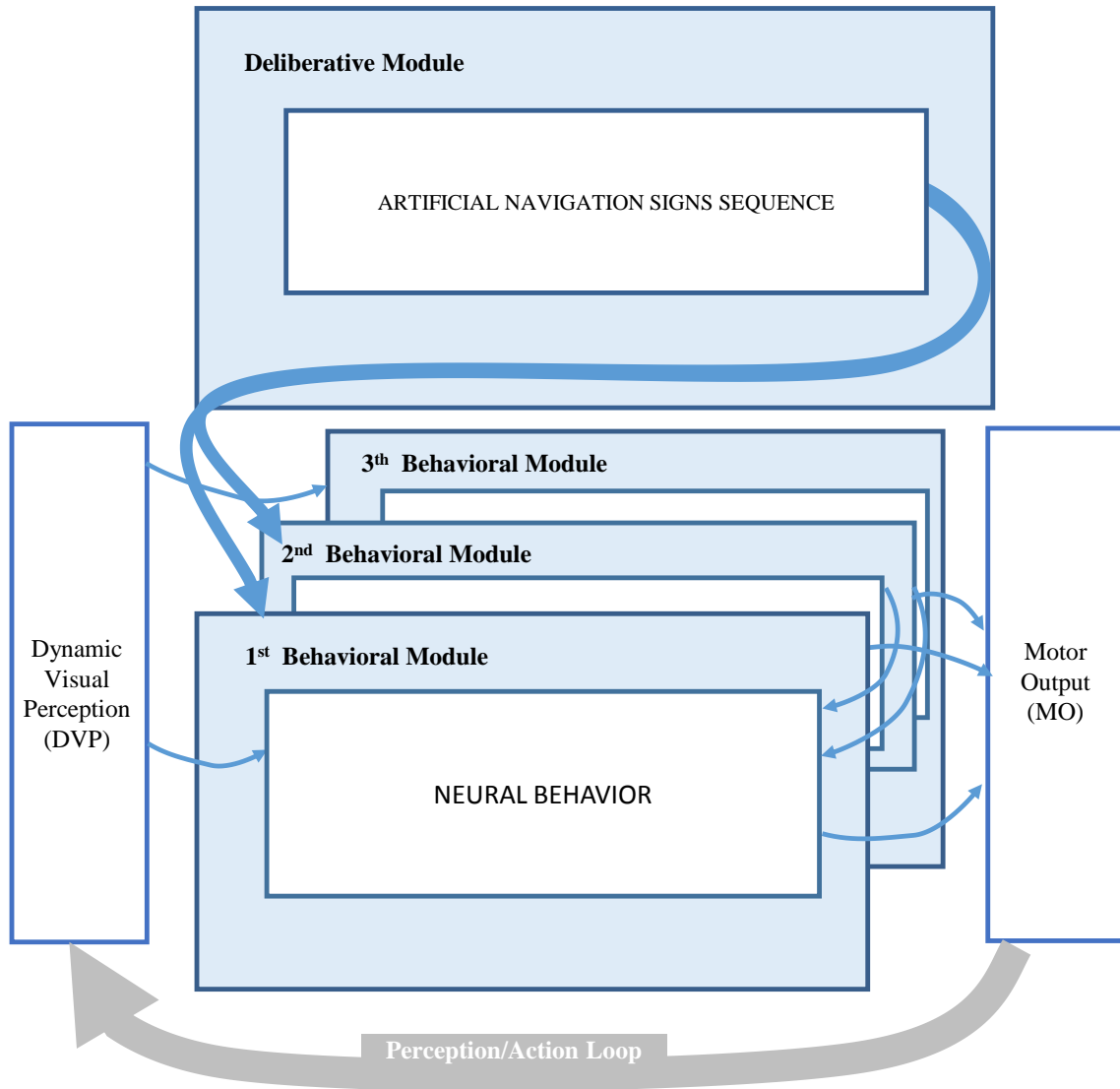


Figure 1.3: RHIZOME Architecture

The first level, Reflex behavior box on figure 1.4, uses a reflex mechanism that controls directly the robot's actions based on the information extracted from the perceived input. The second level, Recognition box on figure 1.4, uses a cognitive mechanism performing recognition by integrating the aforementioned perceptive flow and learning sensory-motor associations. Figure 1.4 shows the PerAc architecture(left) which is used in the behavioral module of the Rhizome architecture (right).

Thanks to the generic composition of the proposed RHIZOME architecture, it is possible to develop the architecture further with respect to robustness and completeness by building new layers and modules separately and simply adding them without modifying the already

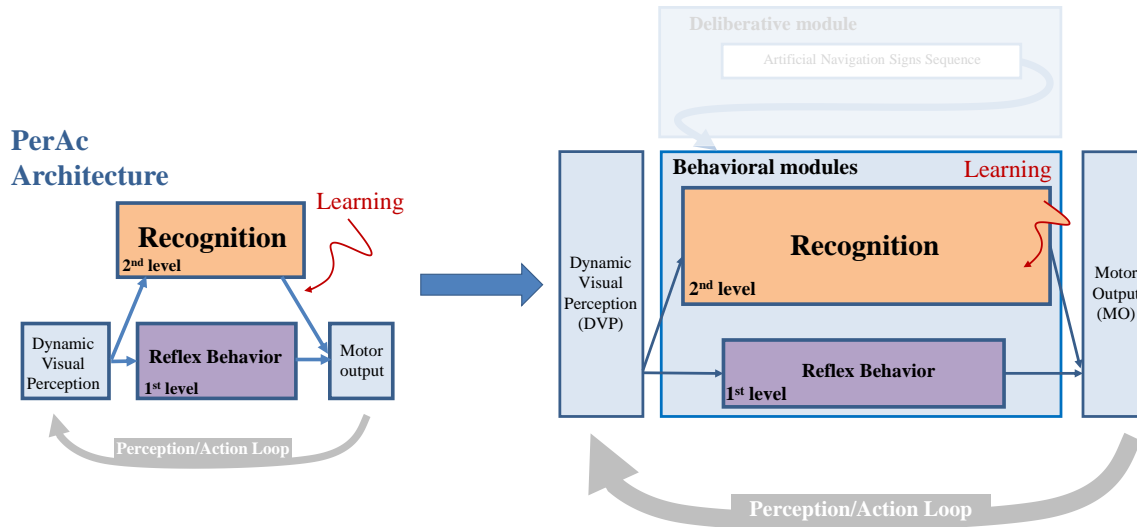


Figure 1.4: PerAc architecture(left) [Gaussier 1995]used in the behavioral module of the RHIZOME architecture (right).

in-built components or modules. One can certainly go further and further on the construction of many modules, as long as there are always new scenarios constraints to overcome.

In the context of this work, the RHIZOME architecture was conceived, built and implemented through three different scenarios under which, three interdependent architectures emerged, each responding to the different scenario constraints.

Deterministic scenario

- Rhizome1: Exploring the world with little information
- Rhizome2: Map-using autonomous navigation

Stochastic scenario

- Rhizome3: Self-learning and adapting according to unforeseen changes

The architectures should not be regarded as if there was a hierarchy among them or as if they followed an evolution pattern where each architecture is the improvement of the previous one. Conversely, starting from the simplest scenario imagined, the emergence of the first architecture occurs. Then, its functionality together with its components propels the functionality of a second one by integrating new components, and thus, the entire ensemble of both architectures propel the functionality of the third one. Each architecture is as important as the others are according to its corresponding scenario.

1.4 Limitations

In order to accomplish a project on autonomous robot navigation, it is necessary to take into account from the start the requirements that the task involves within the given constraints. This implies defining beforehand the scope and limitations of the project. While the scope has been presented all along this chapter, some of the most important tasks not considered in this work are presented below.

Even though, localization is an important module in the achievement of a successful navigation, it is not explicitly considered within the scope of this work. However, it is assumed that by the use of *a priori* information of the world (artificial navigation signs), the position of each navigation sign within the environment is implicitly known and thereby the position of the robot can be globally obtained. Likewise, in the absence of artificial signs in the environment, the use of vision techniques used in this work allows the robot to be locally localized. For a better insight of these techniques the reader can refer to the state-of-the-art Chapter 2 section 2.4 where each navigation strategy is explained in terms of the techniques used for localization and planning based on visual perception.

Another important task to consider in any navigation mission is the management of obstacles (both static and dynamic) which represent a research subject on its own in the robotics field. Obstacles avoidance is usually tackled by employing a variety of ranging sensors. However, since the primary goal of this work was to use a camera as the only sensor, the obstacles avoidance task is not considered in this work.

It should be noted that the author is wary that these limitations along with many other **can and must be** considered in the development of the architecture for future work. For instance, the consideration of new scenarios where the signs may be substituted with other relevant visual or non-visual cues requiring the use of different and multiple types of sensors.

1.5 Outline of the thesis

This thesis has been organized in three parts as follows:

PART I: STATE-OF-THE-ART

Chapter 2 describes the state-of-the art of vision-based robot navigation in terms of the functional modules: visual perception, world modelling, localization and path planning. More precisely, it describes their implication in each different type of navigation strategies by following a transversal structure proposed by the author. Two types of navigation strategies are here distinguished: mapless navigation and map-based navigation composed itself of map-using and map-building navigation.

Chapter 3 presents a detailed description of the state-of-the-art of the currently existing control paradigms. The author has chosen to present them according to two different viewpoints that can, of course, represent a single architecture. On one hand, the functional viewpoint classifies the paradigms in terms of their internal functionality and thus in terms of their capabilities to act on the environment when performing a given task. On the other

hand, the design viewpoint specifies two paradigms based on how the data information is processed and propagated through the systems as well as how the knowledge is ordered. The common misconceptions found in the literature are here cleared and a summary of the advantages and disadvantages of all paradigms is given.

PART II : THE RHIZOME ARCHITECTURE

Chapter 4 explains the foundation of the RHIZOME architecture by explaining the main components of RHIZOME 1. Since the navigation signs sequence is directly given to the robot through a command program, this chapter focuses mainly on the implementation of the different layers composing a first behavioral module, which integrates the given sequence and uses it to allow the robot to navigate towards its final destination.

Chapter 5 introduces RHIZOME 2 by presenting a thorough document analysis process of a floor plan of a building allowing the robot to extract by itself the sequence of signs together with the corresponding directional meaning each sign denotes. A second behavioral module is added to the architecture for the integration of the directional meaning into the system.

Chapter 6 describes in detail the process allowing the robot to learn and recognize a place based on natural navigation signs (patterns) and their relative positions perceived in its surrounding. RHIZOME 3, implements the place recognition system in a third behavioral module that together with the other modules presented in chapter 4 and chapter 5 allows the robot to navigate autonomously by coping with unforeseen situations.

Each of the above chapters presents a set of experiments and results validating the advantages and feasibility of the proposed approach. All experiments were carried out within the same environmental constraints and navigation conditions in order to, not only; evaluate the functioning of each of the architectures but also to allow the possibility of distinguishing the functionality of each architecture with respect to the others. A discussion of the results opening to new perspectives is also given.

PART III : GENERAL CONCLUSION

Chapter 7 concludes the thesis, providing a summary of the presented research and giving an outlook of the future challenges for autonomous mobile robot navigation. It also discusses the anatomy of the RHIZOME architecture as a multi-hybrid architecture attempting to conciliate most so-far used paradigms in the navigation task of mobile robots.

Part I

State-of-the-art

Vision-based robot navigation

Contents

2.1	Introduction	21
2.2	Visual perception	24
2.2.1	Introduction	24
2.2.2	Pattern description and representation	26
2.2.3	Pattern recognition	29
2.2.4	Place recognition	32
2.2.5	Vision in robot navigation	44
2.3	World Representation	47
2.4	Localization and Path planning	49
2.4.1	Introduction	49
2.4.2	Map-based navigation	50
2.4.3	Mapless navigation	59
2.4.4	Summary of the types of navigation	63
2.5	Conclusions	64

2.1 Introduction

Vision-based robot navigation was proposed by Desouza [DeSouza 2002] as being structured in two main topics regardless the vision sensor used in any system: «outdoor» navigation and «indoor» navigation.

On one hand, outdoor navigation is classified by the regularity or not of different properties in the environment: structured and unstructured. While in structured environments, the navigation task can be performed consistently by detecting and following the lines of the road, paved paths, or others[Rasmussen 2014], in unstructured environments, the navigation can be a more complex task to achieve as no regular properties can be tracked. In this case, the robot needs to either explore the vicinity of its environment in a random way or have a fixed goal position by using a map of the area and perform a localization algorithm to execute its mission [Jiang 2013]. However, since outdoor environments can be large in size and extremely irregular, the computational resources, time and storage capabilities required, might be fairly huge.

On the other hand, indoor navigation is subdivided into map-based, map-building and mapless navigation. While map-based navigation provides a model of the environment before the navigation starts, map-building builds one as the robot navigates the environment. Mapless navigation systems instead, do not employ any representation of the world and the movements of the robot depend on the elements observed in the environment. Following the same line of this taxonomy,[Güzel 2013] presented a survey of mapless strategies for autonomous vehicle vision-based indoor navigation as illustrated in figure 2.1.

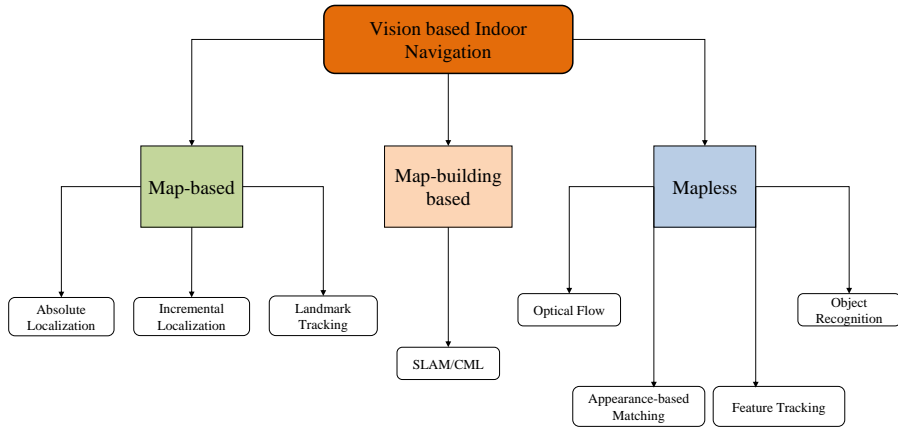


Figure 2.1: Vision based indoor mobile robot navigation techniques proposed by [Güzel 2013]

Such indoor and outdoor navigation distinction has been used as a referent in robot navigation research for several years. However, with the continued progress on sensors as well as mechanical and control aspect of mobile robots systems, several works hold into both categories outdoors and indoors, especially those using a map.

Consequently, the state of the art presented in this work makes no distinction on the type of the environment to build the system, but instead; it focuses on the distinction of the systems needing a representation of the environment to navigate the working environment (**map-based navigation**), and those that do not (**mapless navigation**). A comparable classification has been presented in a survey by Bonin-Font [Bonin-Font 2008], which gives a detailed description of map-based and mapless navigation systems.

Additionally, these types of robot navigation include a combination of the following interconnected functional modules:

Perception: Perception provides the input for a successful control, decision-making and interaction with other agents (robot, humans) in the environment. Perception is the process of interpreting and transforming the sensory information of the state of the robot, the environment and other external entities into a representation that can be used for further processing or further actions.

World representation: By the use of different sensors, it is possible to represent the navigation world into a map and directly use it for computing the path trajectory to execute the navigation task. It can also be possible to post-process it to update new information for

better accuracy (mapping), and thus, achieve a more precise localization. Such map representation can either be built in advance or constructed as the robot discovers its environment. In either case, the representation can be divided into two categories resorting to either *metric* or *topological maps*.

Localization: The localization of the robot denotes its capacity to establish its own position and orientation in the environment. Localization techniques need a certain knowledge of the environment, usually the origin or destination point or a map. Several techniques have been proposed and their difference depends mainly on the nature of the robot's sensors, the environment and the initial available information.

Path planning: Path planning is an extension of the localization task, in which it is necessary to determine the starting and final goal position of the robot, within the same reference system, in order to plan an optimal collision-free path amidst obstacles in the environment (i.e. walls and objects) and thereby navigate towards the final destination. The criterion of optimal performance depends on the application required. It can be chosen in terms of distance (shortest path), time (fastest) or energy (least energy consuming). Path planning algorithms are measured by their computational complexity. It can be divided into two categories based on the availability or absence of a complete representation of the world, namely *off-line* and *on-line*.

The implication of these functional modules in each type of navigation strategy are described in this section by following a transversal structure, which is best understood by the figure below (figure 2.13). Whereas both types of navigation need the perception module to capture and understand the information of the environment, only the map-based navigation requires the world-representation module. Similarly, in order to navigate from a starting point to a final destination, they both need to find ways of localizing in the environment and plan a path trajectory.

Hence, this section comprises of three parts describing the figure 2.13.

- Firstly, an overview of the techniques referring to **visual perception** is given. It refers to a branch of machine learning known as pattern recognition that focuses on the recognition of patterns. This latter is used in the different navigation strategies for detection, matching and recognition of various landmarks in the environment. Additionally, place recognition, which is mainly used by mapless strategies is also described. A review of vision-based sensors used in robot navigation is given.
- Secondly, a description of the **world representation** functional module is given. However, this section is rather small compared to the other two since most of the techniques allowing the construction or utilization of it are related to the localization and type of navigation strategies, which are explained in the third section. Therefore, only the description of two of the most common type of maps (metric and topological) on which our work is based is here given.
- Finally, since the **localization and path planning** functional modules depend mainly on the availability or absence of information provided in advance such as a map, the different techniques employed are explained in terms of each type of navigation strategies.

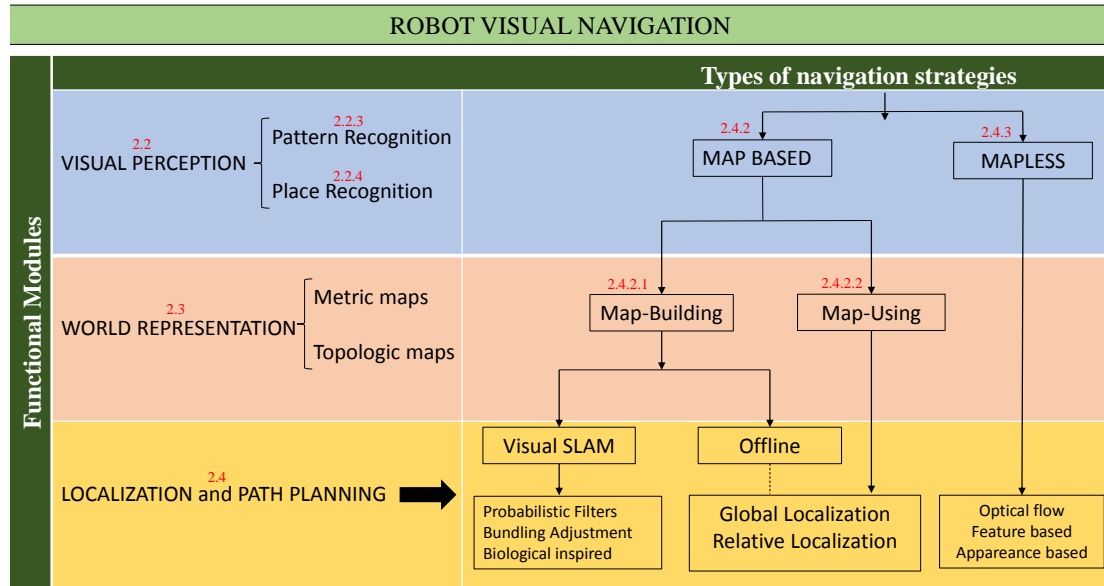


Figure 2.2: Transversal structure representing the implication of different functional modules in all types of navigation strategies.

2.2 Visual perception

2.2.1 Introduction

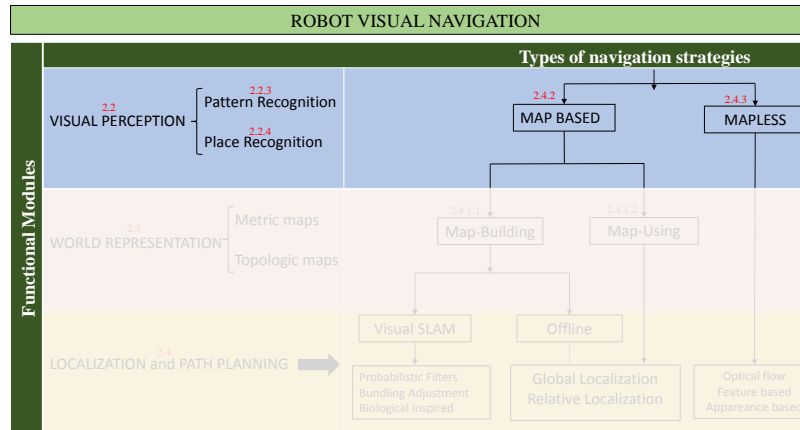


Figure 2.3: Visual perception. Transversal structure representing the implication of different functional modules in all types of navigation strategies.

Robot navigation based on visual perception systems (such as onboard camera systems) has been especially prevalent over the last three decades. These systems are robust and reliable as they provide detailed information about the environment, which may be overlooked

by other types of sensors.

In the context of robot navigation, the robot needs to be endowed with the capacity to analyze its surrounding environment. Therefore, the use of pattern recognition algorithms is essential for either modelling the environment or detecting and recognizing landmarks (usually used for references purposes) depending on the type of navigation performed by the robot.

The pattern recognition problem is an essential research topic in computer vision. Most of visual tasks in applications such as robotics rely fundamentally on the capacity to recognize patterns, which permits to recognize faces, objects, places, complete scenes, etc. The recognition process can be described as a set composed of two different processes (see figure 2.4). First, the acquired images undertake a description and representation processes of different patterns which allow to simplify their learning and distinct recognition at a later stage. Certainly, before carrying out these processes, it is necessary to perform an image pre-processing, similarly to most of image processing tasks. Image pre-processing is essential for removing any problem in the image related to the acquisition of it, such as illumination, noise, perspective distortion in order to allow improving the quality of the image.

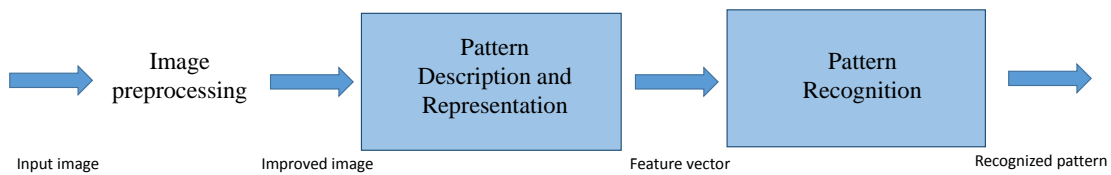


Figure 2.4: Pattern Recognition process composed of image processing, followed by a pattern description and representation process and finally a pattern learning and recognition process. Image processing is not detailed in this work

Furthermore, a place can be also identified and used as a more stable reference point to be recognized by considering a set of patterns placed at different location within the panoramic visual field of the robot. This same mechanism has also been seen in some insects like desert ants, which make use of a visual spatial memory to return to their nests, foraging stations or other. For instance, Wehner and Raber [Wehner 1979] showed in an experiment that when leaving their homes, the ants take snapshots of the patterns around their nests and keep in memory their location. Then, they look for the same patterns located at the same position and by means of a correlation method; they are able to return to their nest.

Based on this insight, a robot could learn a place by keeping in memory the location of the most relevant patterns perceived around itself, and then easily return to it when necessary by recognizing the same learned patterns as presented in this work.

Hence, this section describes the two processes of pattern recognition allowing visual recognition with an emphasis on the methods used in the development of the work of this thesis. Additionally, it presents the place recognition state-of-the-art from the robotics point of view detailing a biological approach. Finally, an overview of different sensors used in

vision-based robot navigation is given.

2.2.2 Pattern description and representation

Features define the relevant parts that differ from their immediate neighborhood in an image and the detection of which can give a cue about the possible existence of any pattern within the image. They can be found as isolated points, continuous lines, or small patches depending on the model applied. A complete description of each of these features can provide information about a pattern and thus help to identify it as such. Hence, it is important to provide a sufficiently detailed description so that the features can be recognized under different circumstances like changes in images scale, perspective, noise and illumination.

To that end, different features description algorithms, attempting to describe the features as precisely as possible to be recognizable at a later stage, have been proposed. However, even though, they have shown to be very robust to any form of variations, most of the local descriptors are high-dimensional and the computational cost of matching their similarity within a large database is quite high. Therefore, the bag of visual words has been proposed as an alternative to mitigate this problem. Thus, instead of directly search the similarity between the descriptors, it quantizes the feature space of local descriptors into discrete «visual words» (clusters) and the matching can be easily performed by simply counting the features assigned to each cluster.

A particular advantage of this representation is that it fixes the dimensionality in all images, which ease the work required for most machine learning that assume by default a vectorial space input.

As follows, we give an overview of the most common features detectors and descriptors proposed over the past years in the literature followed by a detailed explanation of the SIFT algorithm [Lowe 2004] that is used in the context of this work to detect and describe local features in images. The choice of the SIFT descriptors over the others was made by considering its robustness to handle viewpoint variations and its highly distinctive description for reliable matching while being fast at extracting the local features. We end up with a detailed description of the bag of visual words.

2.2.2.1 Feature detection

Three categories can be manly distinguished: Edges detectors, corner detectors and blob detectors.

Edge detectors: This type of detectors aim at detecting a set of points forming curved line segments in the image, which correspond to a sharp change in the intensity of luminosity in a grey-level image. It usually designs a boundary between two images regions, which allows filtering out the non-relevant information of the image, thus reducing the amount of data to be processed. Different detectors have been proposed in the past. The canny edge detector [Canny 1986], is one of the most strictly defined methods that provide good and reliable detection. The Deriche edge detector [Deriche 1987] based its algorithm on Canny's optimal criteria for edge detection, therefore it is often referred as the Canny-Deriche detector. The

difference, though, lies on the implementation on the first two steps out the four proposed by Canny. In fact, it uses an IR filter that optimizes the canny criteria given by its facility to adapt to the characteristics of the processed images using only one parameter. Sobel and Fedelman [Sobel 1968], proposed a filter that convolves the original image with a two 3×3 kernels in order to calculate the approximations of the derivatives in horizontal and vertical directions allowing to reduce costs in terms of computation.

Corner detectors: Corners are regions in the images with large variation in intensity in all directions. Therefore, these type of methods attempt to detect interest points corresponding to double discontinuity of the intensity function, produced by the intersection of two edges, the reflectance discontinuity or depth discontinuity. The best known and most used corner detectors is the Harris detector, which is an improvement of the method proposed by [Moravec 1985] that finds the presence of a corner by computing the similarity of a patch centered on every pixel in the image with other overlapping patches. Harris and Stephens proposed to find the difference in intensity of the corner score with respect to all directions, instead of using shift patches as Moravec's corner detector did. In an evaluation carried by Schmid [Schmid 2000], the Harris corner was proved to be the strongest and most informative detector. Edward Rosten and Tom Drummond [Rosten 2006] proposed the FAST (Features from Accelerated Segment Test) algorithm as a solution for faster corner detection in real time applications. Even though, it is much faster than other existing corner detectors, it is limited by its threshold dependence and by the lack of robustness to high levels of noise.

Blobs/region of interest Blobs can be defined as regions of the image that are lighter or darker than their surroundings. These detectors aim at extracting all the points of interest that are inside the blobs and which are considered to be similar to each other. Two classes can be mainly distinguished according to the position on the image: the differential methods, which are based on derivatives of the function with respect to position, and the methods based on local extrema, which are based on finding the maxima (lighter regions), and the minima (darker regions) with respect to their neighborhood. One common blob detector is based on the Laplacian of the Gaussian (LoG) which convolves the original images with a Gaussian kernel. Similarly, another approach was proposed and is referred as the difference of Gaussians (DoG) approach where blobs can be detected from scale-space extrema of differences of Gaussians. The difference compared to the LoG approach can be found in [Lindeberg 2012]. Mikolajczyk and Schmid [Mikolajczyk 2004] proposed a hybrid operator between the Laplacian and the determinant of the Hessian blob detectors, where spatial selection is done by the determinant of the Hessian and scale selection is performed with the scale-normalized Laplacian.

2.2.2.2 Feature descriptors

A feature descriptor can be defined by a set of scalar numbers generated to describe an object [Erusk 08]. In other words, a signature is built representing the contents of a region in the image. Mostly all object recognition systems use descriptors to describe the regions of interest. However, the choice of features is complex and depends on several factors such as the

class of the object in question, the characteristics sensor, the context and the task to achieve. The choice is often based on a compromise between the accuracy and the generality of the features. Indeed, in the field of object recognition, it is necessary to find a characterization method, which extracts the most effective local descriptors for generic recognition. The corner detectors described above are usually rotation-variant which means, even if the image is rotated, the corners can still be detected. However, they do not handle the problem when the image is zoomed and what it was supposed to be a corner is no longer one but a flat curve.

Therefore, in order to overcome this problem D. Lowe [Lowe 2004] came up with a new algorithm called SIFT(Scale-Invariant Feature Transform), that in addition to its partial invariance to affine distortion and illumination changes, it is invariant to scale which allows to robustly identify objects even among clutter and partial occlusion.

Likewise, the SURF (Speeded-Up Robust Features) algorithm was later proposed by [Bay 2006] as a speeded-up version of SIFT. They differ from the fact that SURF finds scale-space by approximating LoG with Box Filter whereas SIFT approximates Laplacian of Gaussian with Difference of Gaussian. Additionally, it uses the sign of Laplacian (trace of Hessian matrix) for underlying interest point, which permits to distinguish bright blobs on dark backgrounds from the revers situation. Even though, SURF has proven to be at least 3 times faster than SIFT while being good at handling blurred and rotated images, it is not good at handling illumination and viewpoint changes.

Despite the fact that these two methods are called descriptors, they both also provide a method to find the features, thus, they are also feature detectors.

Later on, the BRIEF (Binary Robust independent Elementary Feature) algorithm was presented [Calonder 2010] as the first binary descriptor. It does not have an elaborate sampling pattern or an orientation compensation mechanism. It does not provide any method to find the features and it proves to be faster for calculation and matching. In 2011, the ORB algorithm was proposed as good alternative to SIFT and SURF in terms of computation costs and matching performance [Rublee 2011]. ORB combines a FAST detector and the BRIEF descriptor with some modifications to overcome the poor performance of BRIEF with rotation.

2.2.2.3 Bag of visual words

The bag of words model was initially developed for text categorization [Lodhi 2002], where each document is represented by a histogram based on the frequency of appearance of each word of the vocabulary. Similarly, it was then applied to image categorization by Csurka [Csurka 2004] in the field of object recognition, where the images are represented by a histogram that counts the number of occurrences of each class of the local representation (features) called visual words by analogy. Moreover, [Nister 2006], [Chum 2007] have shown that this simple but effective representation of images is particularly useful for matching features for specific instances of patterns. By the same line, accurate results were presented in recent object recognition challenges by [Everingham 2008] and [Berg 2010]. Furthermore,[Torii 2013] used the bag of visual words model with a simple modification of weights over a scalable detection method allowing place recognition with repeated structures.

The bag of visual words model consists of two steps. First, the construction of a vocabulary allowing to identify the visual words by clustering the whole set of features extracted according to their similarity. Second, the assignment of features of the new images to the cluster with the closest centroid, followed by the histogram of the number of occurrences of the features in the given image. The main advantages of this method is its simplicity, its computational efficiency and its invariance to affine transformation, occlusion, lighting and intra-class variation. For a deeper insight, the reader can refer to the appendix toolbox section.

2.2.3 Pattern recognition

Pattern recognition is the process of recognizing patterns and regularities in data. The recognition process consists in generating a general function by assigning an output value to input data based on key features. The function is a result of a learning-by-example algorithm, which allows predicting reasonable outputs for new unseen data input by taking into account the statistical variation.

Depending on the given task and the availability of the examples dataset, the assignation of output values to input data can be done in advance during a training phase. Thus, learning is performed offline and recognition of new data is performed online. This process is called traditional or batch learning which opposes to incremental learning [Geng 2009] or adaptive learning [Carpenter 1987] where the learning and recognition process needs to be executed simultaneously and adapt itself to new input data.

Incremental learning is necessary when the complete input dataset is not available at once, but instead the dataset appears as the system evolves according to the requirements of the task. Pattern recognition has applications in computer vision, radar processing, speech recognition, and text classification. Additionally, it is generally categorized in two types according to a learning procedure: supervised and unsupervised learning.

Figure 2.5 presents some state-of-the-art algorithms according to the type of learning algorithms used in both batch and incremental learning.

Another type of algorithms allowing recognizing input data is also possible by using pattern matching which is opposed to pattern recognition in the sense that is not considered as a type of machine learning. Therefore, no learning is performed. However, pattern-matching algorithms can sometimes perform an output of similar quality as that provided by pattern recognition algorithms. Thus the common assumption that pattern matching is a technique of pattern recognition.

2.2.3.1 Batch learning

The batch or traditional learning process is accomplished in a sequential mode by following two different but correlated phases: the training and the recognition phase. First, a complete data set is trained in advance (off-line) by a classifier and then the recognition takes place online by taking into account what it has been learnt. The template dataset is considered sufficient to store all the necessary knowledge to be compared to, at a later stage, to new

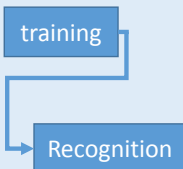
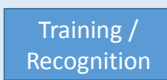
Machine Learning	Description	Representation	Supervised learning algorithms	Unsupervised learning algorithms
Traditional learning (Batch)	A complete dataset is trained in advance by a classifier and is considered sufficient to store all the necessary knowledge to be compared to new input information in order to perform the recognition task.		Support Vector Machines (Cortes, C., & Vapnik, 1995) Artificial Neural Networks (Riesenhuber et al. 1999) K-Nearest Neighbors (Zhang,H et al. 2006) Random forest (L.Breiman 2001) Deep Networks (Ivakhnenko's 1971)	Kmeans (Jurie, F., & Triggs, B. 2005) K-Nearest Neighbors (Ranzato et al. 2007) Mean Shift (Fukunaga, K., & Hostetler, L. D. 1975) Expectation-maximization (Dempster et al. 1977)
Incremental Learning (adaptive)	Training examples become available over time. Therefore, both training and recognition tasks take place at the same time. The recognition system is able to adapt itself to the new incoming information.		SVM (Syed et al, 1999) ARTMAP(Carpenter et al. 1991) Fuzzy ARTMAP(Carpenter et al. 1991)	ART1 /ART2 (Grosseberg and Carpenter 1987)

Figure 2.5: Machine-learning techniques. Supervised and unsupervised algorithms according to traditional learning and incremental learning

input information in order to perform the recognition task. However, it implies that new data set expected to be recognized, has already been learned, in which case, the batch learning techniques generates the best predictor.

2.2.3.2 Incremental learning

Due to the constantly changing and unpredictable environments encountered in many real-world applications where the chances of emergence of completely new elements over the time are quite high, it is unconceivable to assume that a fix and *a priori* training set is sufficient to store all the necessary knowledge to be compared to any new input.

Therefore, the incremental learning process takes place every time a new input is fed to the system and it adjusts what it has already been learned accordingly. Several approaches have been proposed in the literature. For instance, [Syed 1999] stated that the Support Vector Machines properties, which allow summarizing data by preserving the support vectors, are a good indicator to extend their use, usually employed for batch learning, to fit in an incremental learning framework. [Ross 2008] proposed a visual tracking method capable of adapting to the target appearance changes by incrementally learning a low-dimensional subspace representation.

Following the same principle line but employing different terms for the same meaning, [Schuermans 2007] proposed an implicit online learning algorithm that can learn from increasing training examples while still updating its parameter vector to minimize a functional risk. On the other hand, [Huo 1997] refer to an adaptive learning algorithm, which copes with the time-varying nature of some acoustic and environmental variabilities, including mismatches caused by changing speakers, channels, and transducer.

Similarly, Grosseberg and Carpenter [Carpenter 1987] introduced an adaptive algorithm by using neural networks as a solution to the plasticity-stability dilemma that has to be handled in incremental learning algorithms. The plasticity defines the capacity of a system to adapt to a changing environment, however, the system can suffer from instability as it can forget what it has previously learned when learning new information. Therefore, the plasticity-stability dilemma consists in finding how a learning system can still learn new information by preserving its previously learned knowledge. To that end, Grosseberg and Carpenter proposed the Adaptive Resonance Theory (ART), which is a self-organizing competitive neural network.

The basic ART is based on an unsupervised model and has a self-regulating control structure that allows a stable autonomous recognition and learning. It is mainly composed of four components: a comparison vector field, a recognition field, a vigilance parameter and a reset module. Both, the comparison and the recognition fields are composed of a set of neurons encoding respectively the input vectors and the category to which the input vectors are classified. The vigilance parameter works as a threshold of similarity between the input vectors and the categories and the reset module compares the threshold value to the strength of the recognition match after the input vectors are classified. The value of the «vigilance parameter» is quite essential on the recognition task. The memory can be refined or generalized depending on the chosen value. Hence, a higher value produce the creation of many categories whereas a low value results in fewer categories. For a better insight of this theory the reader can refer to the appendix section A.2.3.

2.2.3.3 Learning procedure types

Pattern recognition is mainly classified into two categories according to the nature of learning procedure used to produce the output value. In supervised learning, the input dataset is trained according to a desired output dataset that is provided. It is said that the example dataset is labeled, whereas in unsupervised learning the example dataset is not labeled since no desired output dataset is provided. Therefore, the input dataset is clustered into different groups.

There exist also a combination of both categories known as semi-supervised learning which usually combines a small set of labeled data with a large amount of unlabeled data. It starts with labeled examples and then predicts the output of unlabeled data while using their statistical distribution.

Supervised learning: These algorithms analyze the training data and determine the class labels of new data. The training data consists of a set of input data properly labeled by hand with a desired output. Then, the train data is used to produce a function that attempts to

allow mapping new data while generalizing as accurate as possible to new unseen data. In other words, it has to provide a correct output when new unlabeled data is given as input. These methods are usually fast and accurate when new data has already been learned.

Unsupervised learning: During the training phase in unsupervised learning algorithms, the model or the input dataset is provided without the desired output which means that the examples are unlabeled. Therefore, it attempts to find inherent patterns in the dataset that can help it to determine the correct output value in new datasets. The input dataset is clustered into different groups or classes on the basis of their statistical properties. Then according to some feature similarities, new input data can be associated with one of the created classes. In some cases, there may be no training data at all; in other words, the data to be labeled is the training data.

2.2.4 Place recognition

Different approaches of visual place recognition have been used as important tools for solving mapping and robot localization problems, which are essential in the context of autonomous robot navigation. They endow the robot the capacity of understanding its surrounding environment, knowing its position with respect to a reference point and thus creating a spatial representation of it allowing to ease its navigation task.

A place can be identified as a stable reference point that can be learned by keeping in memory the location of the most relevant perceived patterns within the panoramic visual field of the robot. Thus, recognizing the place consists in recognizing the same learned patterns.

Therefore, many researchers in computer vision have attempted to tackle the visual place recognition problem by retrieving images of the scene, train them and compared them to other images thereafter. The undertaken process for scene recognition can be considered as a more generalized version of the pattern recognition task. Certainly, an image of a given place can be described by the various patterns it comprises of. Therefore, the place recognition task can be solved by using any of the approaches described in the above section, and can be distinguished in two types of recognition depending on the application: a topological place recognition and place categorization.

In the context of robot navigation, a topological place recognition consist in endowing the robot with the capability of recognizing previously observed places in known environments. State-of-the-art visual Appearance-based SLAM (ASLAM) techniques such as FAB-MAP [Cummins 2008], [Ho 2007], [Eade 2008] convert the images from a set of local features, into a bag-of-words representation in order to match the appearance of the current scene to the trained data. This task is quite challenging as the algorithms attempting to match images must cope with problems such scalability, illumination conditions, different viewpoints etc.

For instance, [Knopp 2010] present a method that detects and removes automatically objects that occur at many places and hence they are not representative for any particular place and moreover can lead to confusion when comparing different images. A similar problem is found in images with repetitive structures such as building facades, fences or road markings as highlighted by [Torii 2013]. In their work, they use a robust detection of repeated images and describe a suitable representation for scalable retrieval allowing a better place recognition

performance. [Krizhevsky 2012], [Donahue 2014] and [Sharif Razavian 2014], suggest that a good performance can be achieved by using Convolutional neural networks (CNNs) in the classification task. However, this approach is limited by the huge quantity of training data required.

Place categorization instead, allows the robot to give a semantic labeling to the places by classifying different locations of a new environment into categories such as “office”, “kitchen”, “corridor” etc. in the context of indoor environments. For instance, [Ramos 2012] consider the world as a set of places where each of them has a probabilistic representation learned from images and which are labeled by using a classification procedure. The place recognition is treated as a Bayesian learning problem and it is performed without the need of a map and by using only few training images (usually 3 to 10 per place). [Wu 2009] instead, predict the semantic category of a place from the measurements of the acquired images collected of the spatial location, as opposed to the geometric or topological characteristics.

In contrast, [Nguyen 2013] employ a topological representation of the environment by encoding the neighborhood relations. The connections are related by arcs indicating their spatial relationship. It employs a Fast Learning Artificial Neural Network (KFLANN) as the core unit of the quantization module, which compared to popular clustering methods such as k-means, produces a consistent number of stable centroids, as it is less sensitive to data presentation ordering.

Furthermore, other algorithms such those found in [Fazl-Ersi 2012] and [Ullah 2008] attempt to tackle both problems by providing strong discriminative control for place recognition, while offering a substantial level of generalization for place categorization.

2.2.4.1 Biological approach

Biological systems attempt to tackle the place recognition problem by proposing models which emulate similar behaviors seen in living organisms based on allotethic information when performing goal-orientated navigation tasks [Burgess 1994], [Brown 1995], [Guazzelli 1998], [Redish 1997] and [Filliat 2002].

Certainly, many studies on insects like bees, ants and wasps have shown that they use visual information to return to their nests or to locate a foraging station and then go back and forth between that source and their home [Cartwright 1983], [Gallistel 1993], [Judd 1998]. After several experiments, it has been observed that they store multiples views of a place from different positions in order to learn the place. Then, by comparing and matching the stored images to the newly perceived, they are able to recognize the place.

Similarly, neurobiological studies in mammals like primates and rats have revealed that they also use surrounding visual cues in order to achieve a particular place comparable to insects. However, mammals show higher generalization capabilities and more complex processing when performing recognition of a scene or a place.

Edward Tolman, studied how animals learn to navigate in the environment. He was the first suggesting, from purely behavioral experiments, that local navigation in rats was guided by an internal map or «cognitive map» as he called it; resulting from the exploration of the environment and the relationships between places and events [Tolman 1948]. This suggestion was later confirmed when experiments conducted by [O’Keefe 1971] led to the

discovery of pyramidal neurons in the rat hippocampus (CA3 and CA1 sub regions) that fire at their maximal activity when the animal is at a particular location in the environment and decreases as it goes away from it. They are called place cells and the regions at which they fire at are called place fields, which are almost similar to the receptive fields of sensory neurons.

Hence, O’Keefe and Nadel, inspired by Tolman, proposed years later, the place cells as the basic elements of a spatial representation and the hippocampus as the locus of the cognitive map [O’keefe 1978b]. In fact, the firing activity of the neighboring place cells at different areas thorough the hippocampus as the rat explored the entire environment was an eminent indicator that the place cells enable the estimation of the animal’s current position and are related to the construction of the environment spatial representation [O’Keefe 1976], [Jung 1993]. However, the topology of the environment is not preserved as two place cells may fire for two far away locations in the environment. Additionally, even though the same place cells participates to the spatial representation of different environments, the mutual relationship of the place fields remains unique for each environment [O’keefe 1978a]. Thus, since their discovery there is plenty evidence that stablishes nowadays that the hippocampus plays an important role in the spatial representation of the environment of a great number of mammalian species [O’keefe 1978a], [Rolls 1999], [Ekstrom 2003], [Ulanovsky 2007].

For instance, in [Arleo 2000], [Arleo 2001], the autors showed how place cells and head-direction cells firing enables rat mapping and goal navigation within the arena by estimating the rat’s position as well as its orientation. Later on, based on their previous work, the same authors presented a more refined work by combing allothetic (visual) information and idiothetic (path integration) signals at the level of the hippocampal representation in order to remove singularities caused by perceptual aliasing and solve the hidden-state problem [Arleo 2004]. They employ the Gabor-based decomposition technique as well as the retino-topic image sampling to process visual information.

In [Gaussier 2002], their hippocampal model cells do not code for places but instead for transitions between states. They suggest that such transition prediction mechanism may be significant for novelty detection and merging planning and sensory. The activity of a place cell is a normalized sum of Pr-Ph cells, which were activated during place learning [Gaussier 2000]. In their work, they propose a navigation strategy consisting in planning routes towards the goal in a topological graph (cognitive map) of the environment where the recognition level depends only on the correct recognition of sub-areas of the image centered on focal features.

Similarly [Giovannangeli 2006a] improve the above work by compressing the *what* and *where* information in the Pr-Ph and prove efficient navigation in both indoor and outdoor environments tested on different robotic platforms (Koala K-Tram, Labo3 AAI, Pioneer 3AT ActivMedia).

Visual cues have shown to be essential in the formation of place fields as well as other allothetic cues (auditory, olfactory and tactile). Place fields are usually unaffected by large sensory changes like removing a landmark or many from an environment. However, they respond to subtle changes, such as the alteration of the shape or color of the object for instance [Moser 2008] or even the rotation of remote visual cues in a given environment inducing to the rotation of the place fields themselves [Muller 1987], [O’keefe 1978b]. Moreover, Place cells

activities have shown stable place fields when the rat is in the dark [Quirk 1990], which suggests that place cells not only rely on allothetic but also on idiothetic information (vestibular) which is used by rats and other animals for path integration during goal-oriented tasks.

Hence, similar models have also been proposed improving the foundation of the above navigation systems by integrating other types of cells, (grid cells) [Dollé 2010], [Alvernhe 2012], [Caluwaerts 2012], [Giovannangeli 2008], [Milford 2004], [Milford 2007], [Milford 2010]. Grid cells were discovered initially in the rat’s Medial Entorhinal Cortex (MEC) [Hafting 2005], [Fyhn 2007] and later on in other mammals [Killian 2012], [Ulanovsky 2007], [Yartsev 2011], [Yartsev 2013] as well as human entorhinal cortex [Jacobs 2013], [Doeller 2010]. These cells are called grid cells because of the hexagonal pattern formed by the spatial firing fields that tiled the environment and have been suggested to implement a path integration-based spatial representation [McNaughton 2006]. These kind of models suggest that grid cells and hippocampal place cells have strong functional interactions allowing a robust navigation capacity.

For instance, [Jauffret 2012] present a model that merges visual and proprioceptive primitives. The visual primitives are represented by the place cells which have been developed previously by the same research team [Gaussier 2000] and [Banquet 2005] whereas the proprioceptive primitives are characterized by the implementation of grid cells from path integration. In their paper, the authors present a model of grid cells based on various modulo’s operator applied on path integration, which is merged with visual cells information. This merging mechanism, which combines allothetic and idiothetic information, is based on a pavlovian conditioning rule. As a result, a robust multimodal place cells is built successfully overcoming the perception ambiguity problem.

Similarly, [Tejera 2013] presents an extension of the model proposed by [Barrera 2008] whose place representation module is composed uniquely of place cells. In contrast, Tejera develop a grid cell neural model, which is added to the original place representation module in order to generate neural odometry and spatial localization. The experiments are performed with the Kepera III robot and are inspired by Morris’ water mazes [Morris 1981], [Morris 1984] (open arena experiments) and prove feasibility of the model for short runs.

As a result, most biological models, attempting to emulate the navigation task performed by insects and mammals as the ones presented above, show a great performance when the robot needs to recognize places and localize itself as it navigates the environment. They also show to be very robust when it comes to overcome unforeseen situations

Prehippocampal PCs-based neural model A robust visual place recognition algorithm needs to combine descriptive, discriminative and generalization properties.

Therefore, in order to capture all these properties Gaussier et al. proposed a model of the prehippocampal systems where “place cells” are learned in the entorhinal cortex (EC) as a result of the recognition of a particular configuration merging [Gaussier 1997], [Gaussier 2000], [Gaussier 2002], [Gaussier 2007], [Giovannangeli 2006a] and [Giovannangeli 2006b]. The model was tested on different robotic platforms (Koala, Labo3, Pioneer AT), evolving in indoor and outdoor environments.

In their model, they consider the perirhinal cortex (Pr) and parahippocampus (Ph) as the

possible places receiving local configurations of different kind of information: *what* and *where*. Indeed, two main streams of information have been identified in the cerebral cortex of the brain of mammals for the visual recognition task [Goodale 1992], [Mishkin 1982]. The temporal regions commonly referred as the « what » pathway and the parietal regions commonly referred as the « where » pathway. The former is in charge of identifying and recognizing the features perceived in the visual scene of the place whereas the latter is involved in the analysis of the spatial location of the same [Burnod 1990], [Gilbert 1983]. These input layers converge on a merging layer (Pr-Ph) coding a local view constellation, which is directly connected to the EC-DG where the place cells are learned, the activity of which is a normalized sum of Pr-Ph cells.

Hence, in order to learn a place by following the two-streams hypothesis, it is necessary to; on one hand, describe the perceived features in a distinctive way and on the other hand, find their respective location within the scene. Therefore, in order to achieve this task, a set of group of neurons has been used (see figure 2.6). The perceived image from the visual scene is divided in sub groups of local views and each local view is associated to a neuron from the *what group* (Pr) which is set as its unique identifier. Similarly, its relative position information is associated to a neuron from the *where group* (Ph). Then, the information of all local views perceived in the panorama and coming from both groups converge on a two-dimensional array of neurons, which keeps in memory the resulting value. As a result, a landmark constellation is formed in the *Pr-Ph group* leading to the learning of a new place by recruiting a new neuron in the *place cell group*(PC).

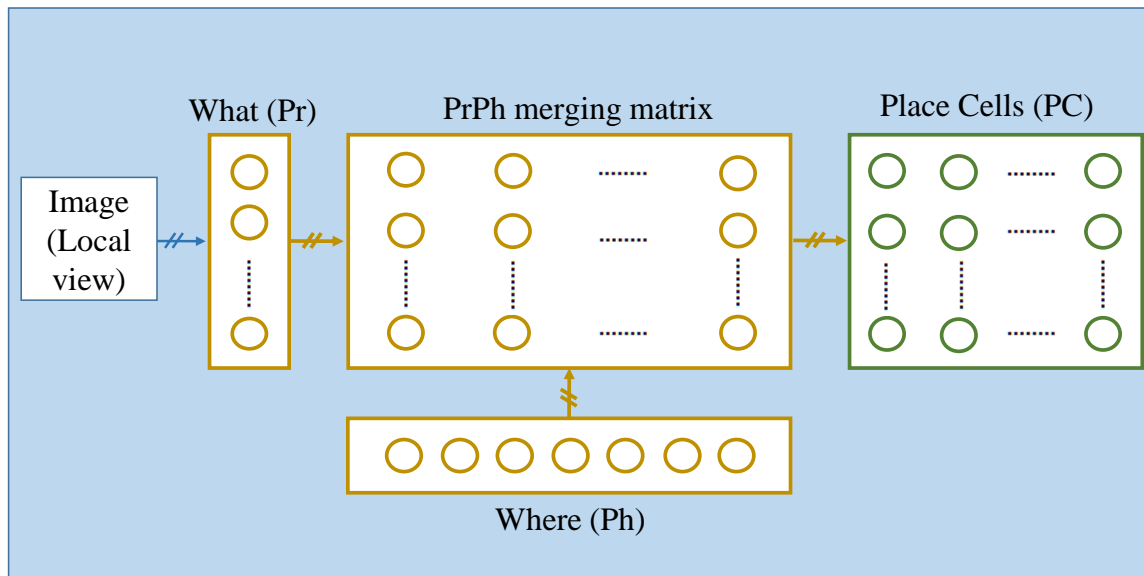


Figure 2.6: Overall view of the place cells model proposed by [Gaussier 1997]

Local view (landmarks) extraction

Network input

The first layer of the architecture extracts autonomously local views from a panoramic image.

In early works, the panoramic image was built from a set of classical images [Gaussier 1995], [Gaussier 2000]. To this end, a servo motor was used to pan the CCD camera which field of vision spanned about 70° . Then, the construction of the global panoramic view consisted of merging only the central vertical bands of each image as the camera distorted the images sides. In total, 24 images were taken per panorama with a 7.5° rotation between each image acquisition resulting in a 250° of field of view. Even though, it was not a complete 360° image, it was enough to prove the robustness of the system in practice. Thereafter, in order to speed up the experimentation an omni-directional CCD using a conic mirror was introduced [Giovannangeli 2006a]. Hence, it was possible to capture in one-shot 360° panoramic images.

Once the panoramic image is obtained, its gradient is used as the only visual input of the system. This process allows to eliminate problems induce by luminance variability likely to appear when performing navigation in real time. The gradient image is then convolved with a difference of gaussiens (DoG) filter in order to detect robust focal points at a particular spatial (low) resolution. In this case, the system focuses on corners and/or edges (see figure 2.7).

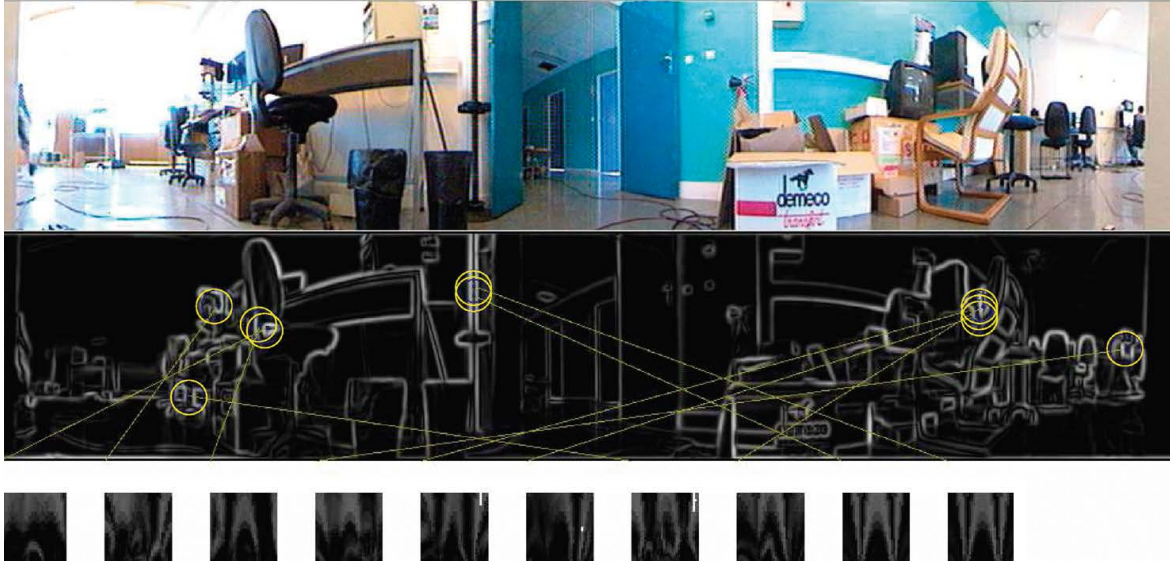


Figure 2.7: State-of-the-art local view landmark extraction. *Top*, the panoramic image taken by the robot. *Middle*, the corresponding gradient picture. Circles represent local area centered on landmark, from which small images are extracted. *Bottom*, small images after the log-polar transform, (Image extracted from [Cuperlier 2007]).

However, in order to reduce the computation time, a simplified process [Gaussier 2000] averaged and weighted all images columns for the points near the center of a column and the resulting one-dimensional signal is differentiated. Thus, local maxima are used as the focal points (see figure 2.8). Finally, a 32×32 -pixel area around each of the focal points is extracted and considered as a local view which undertakes a log-polar transformation so that they can be invariant to small rotations and scale variation.

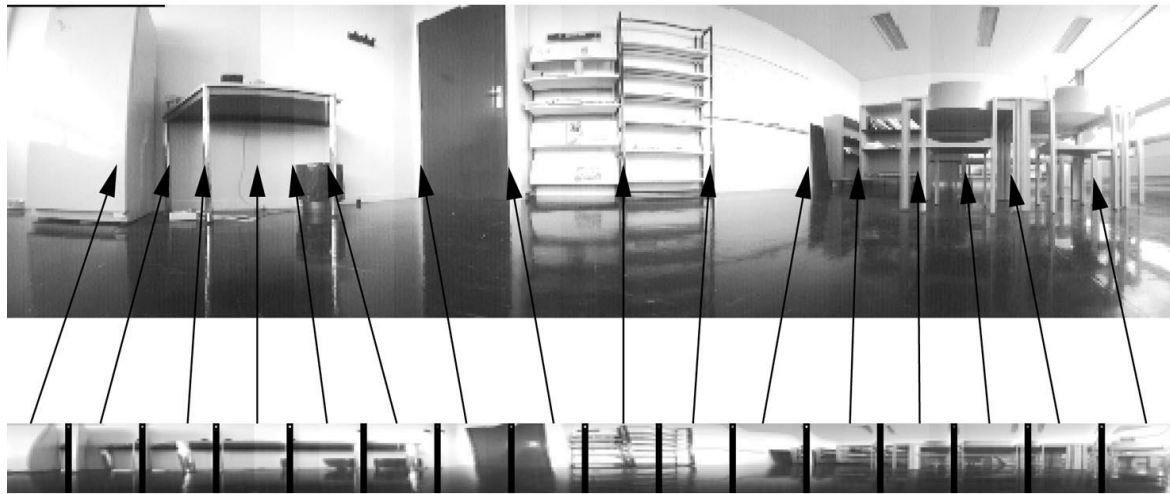


Figure 2.8: 16 examples of 32×32 local views from the panoramic image (image extracted from [Gaussier 2000]).

Landmark and azimuth learning and recognition The extracted local views in the section above, are learned as landmarks when they are seen for the first time. Then the recognition phase takes place by comparing the current local views (potential landmarks) together with their angular positions with the previously learned landmarks. To this end, the *what* and *where* neural groups encode each local view description and azimuth information respectively.

What (Pr) group:

The Pr group is composed of a sufficient number of neurons to encode the total amount of landmarks that can be found in a given exploration environment. Given the lifetime of any robot for learning an infinite number of places in different environments, this number can be considered infinite. In this model, the authors suppose that the visual system can differentiate all the landmarks. Hence, a landmark cannot be found twice in the same panoramic view as it would not succeed in knowing which azimuth is associated to each landmark. Therefore, the same number of landmarks perceived in a given panorama is needed to recruit the **Pr** neurons, and the number increases proportionally to the number of landmarks perceived in new panoramic images.

When learning a local view, the robot recruits one k neuron from the **Pr** group and it associates it to the local view by performing a one-time learning. All weights of the links between the input neurons and the **Pr** neurons are initialized to zero. Then, the synaptic connection weights of a **Pr** neuron are modified (and do not change anymore), according to the following rule:

$$\Delta W_{ij,k}^{Im-P_r} = I_{ij} * R_k^{P_r} \quad (2.1)$$

With $R_k^{P_r} = 1$ when recruited, and $R_k^{P_r} = 0$ otherwise. I_{ij} is the value of the ij^{th} pixel from the local view of the image Im . The recruited neuron is a landmark unit.

The recognition process is performed by computing the norm of the difference between the pixels of the learned landmarks and the current ones. Hence, the activity of the k_{th} landmark unit $x_k^{P_r}$ is computed as follows:

$$x_k^{P_r} = f^{RT} \left(\frac{1}{(X_I Y_I)} \left\| \sum_{i,j=1}^{X_I, Y_I} \Delta W_{ij,k}^{Im-P_r} - I_{ij} \right\| \right) \quad (2.2)$$

With X_I and Y_I the number of pixels on x and y coordinates of the corresponding small local view. $\Delta W_{ij,k}^{Im-P_r}$, the weight of the link from pixel i, j to the k_{th} landmark unit and $f^{RT} = \frac{1}{(1-RT)}[x - RT]^+$ an activation function that extends the dynamical range of the output. RT is a recognition threshold. $[x]^+ = x$ if $x \geq 0$ and 0 if not.

The interest of using the activation function f^{RT} is to allow multiples interpretations of the same local view, which may be perceived from different angles when learning different places. This competition mechanism enhances the built in generalization capability contrary to a Winner-Take-All (WTA) mechanism which prevents place fields from overlapping.

Figure 2.9, shows an example of two landmarks learned as different visual patterns for two different places. Then when placed at an intermediate location place C, both landmark have two different interpretations (high activity value). Hence, the system allows to choose both interpretations instead of only one.

Where (ph) group:

The where information corresponds to the absolute direction (azimuth) of the local view which can be obtained with a compass or any simulation of a vestibular system, such as a gyroscope of inertial system. The group is composed of a limited amount of neurons that encode the landmarks position within the 360 degrees of the panorama view. Each neuron has a preferred direction covering in all the total 360 degrees of the panorama view. Each neuron expresses how near the landmark is from its preferred direction by calculating their angular distance. It follows a strictly monotonous function that decreases from 1 to 0.

The activity of the i^{th} **Ph** neuron is given by the following equation:

$$\theta_i(t) = g^{\rho_1} \left(\left\| 2\pi \cdot \frac{i}{N_{ph^{\theta}}} - \theta(t) \right\| \right) \quad (2.3)$$

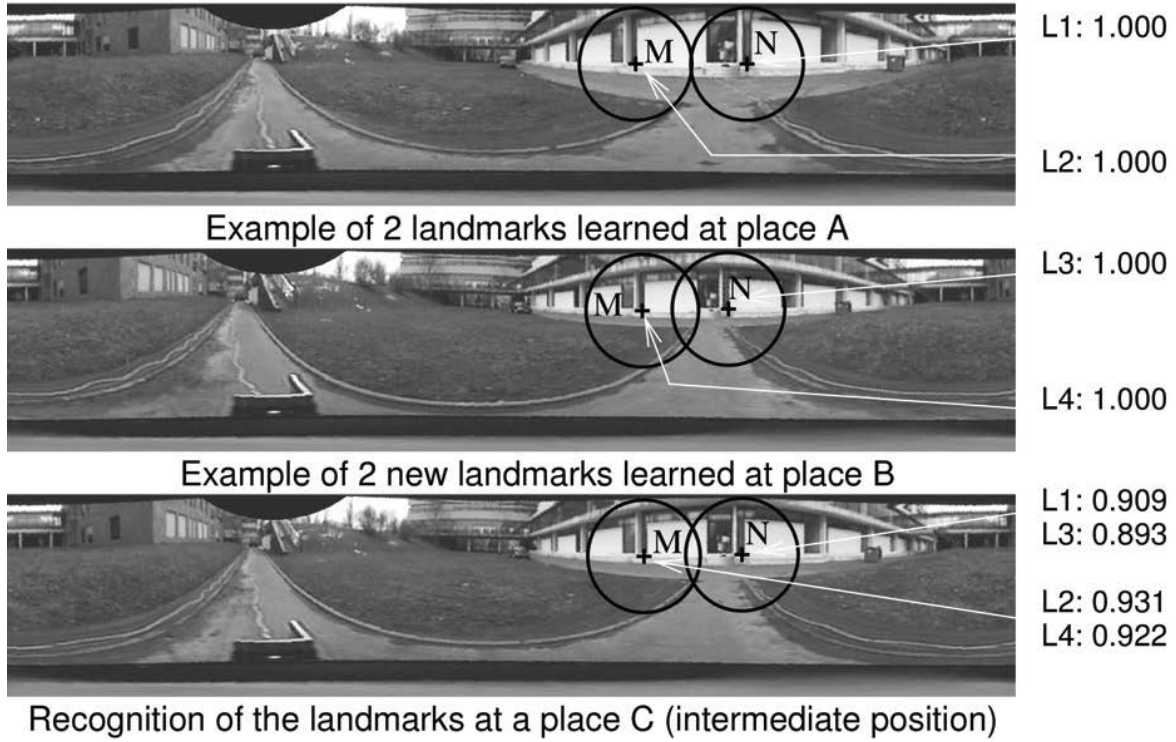


Figure 2.9: Learning and recognition of the same physical landmark by several neurons. The physical landmarks M and N have been learned, for two proximal locations (the two northern crosses), as different visual patterns (upper figures). Hence, in the intermediate location (place C, lower figure), the landmarks have two valid interpretations. In location C, the activity level of «L2 and L4» for the landmark N are rather high and similar as well as the activity level of «L1 and L3» for the landmark M. (Image extracted from [Giovannangeli 2006a]).

With $g^\rho(\Delta_\theta) = [1 - \frac{\Delta_\theta}{\rho \cdot \pi}]^+$ and $N_{ph\theta}$ the number of neurons in \mathbf{Ph} .

The same computation is performed for each neuron in the group with the same θ and since their preferred direction is different, only the closest neuron to θ results with the maximum activity value and consequently gets to encode the landmark position. $g^\rho(\Delta_\theta)$ it is a simple linear diffusion that computes a lateral diffusion around the neuron which preferred direction is the direction of the current extracted landmark $\theta(t)$. Where $[x]^+ = x$ if $x \geq 0$ and 0 if not. Other linear diffusion functions can also be used such as a non normalized Gaussian activity profile [Banquet 2005] as in equation 2.4.

$$\theta_i(t) = \exp - \frac{(2 \cdot \pi \cdot \frac{i}{N_{ph\theta}} - \theta(t))^2}{2\sigma^2} \quad (2.4)$$

As a result, the visual system provides the azimuth of the focus points of the local views and the most activated landmark neurons with their activity level as shown in figure 2.10.

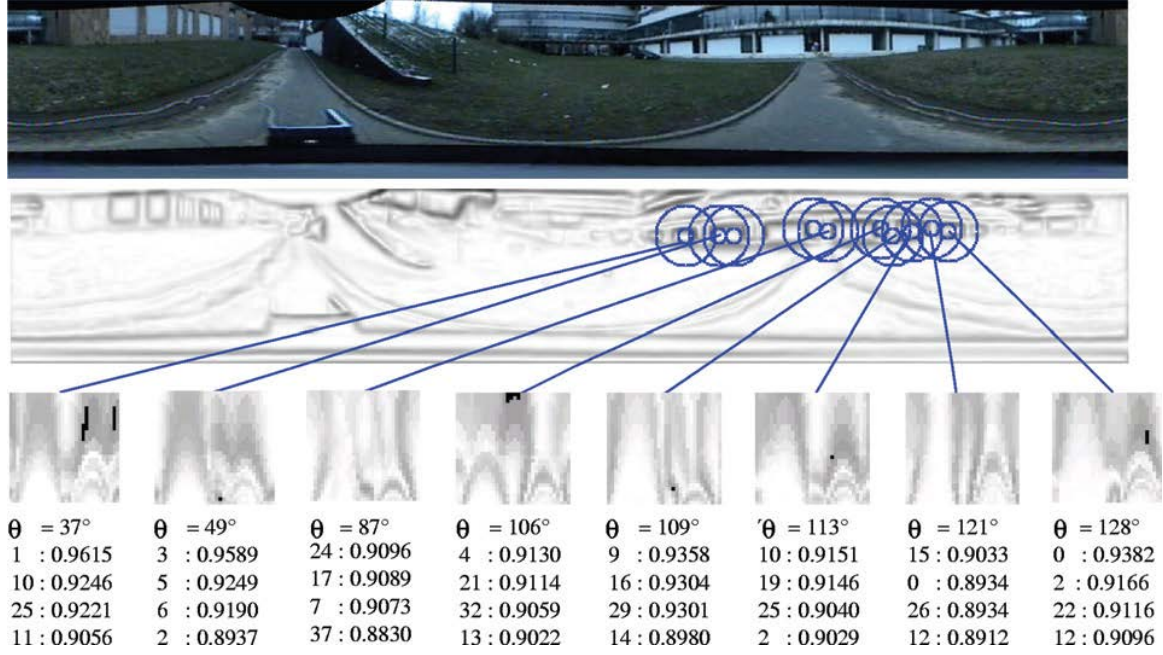


Figure 2.10: Azimuth of the focus points of the state-of-the-art local views. *Top*, Panorama taken by the camera. *Middle*, local views in log-polar coordinates extracted from the gradient image corresponding to the landmarks. *Bottom*, Results of the position of the local views together with the four most activated neurons with their corresponding activity value (Image extracted from [Giovannangeli 2006a]).

Spatiotemporal merging information In order to learn a place, the robot needs to keep in mind the information of all landmarks perceived from its point of view. However, as the analysis of the place which is given by the analysis of the landmarks within the panorama view can only be done in a sequential mode (the system can not recognized several landmarks in parallel), it is necessary to keep in memory the overall information.

Thus, in order to suppress the sequential aspect of the scene exploration, a matrix of neurons stores the information of all landmarks perceived in the panorama view. In fact, the information coming from both **Pr** and **Ph** groups of each landmark converge into the **PrPh** matrix allowing a spatio-temporal merging. As a result, a landmark constellation is formed allowing to learn a new location by encoding a neuron in the *Place cells group*.

PrPh matrix- landmark constellation build-up:

The number of neurons the **PrPh** matrix comprises of, correspond to the number of

neurons allowing to encode as many landmarks as possible within the 360° of view for different learned places. The initial idea was to build a neural matrix of $N_L * N_A$ neurons in which each neuron was linked to one of the N_L landmarks neurons and one of the N_A azimuth neurons. Then the activity of the ij^{th} neuron in **PrPh** can be calculated by computing the product: $S_{ij}^{PrPh} = S_i^L * S_j^A$, with S_i^L the activity of the landmark i and S_j^A the activity of its azimuth j .

However, even though coding such information was correct, it uses too many resources that are not strictly necessary. Indeed, the \bar{n}_a average number of different azimuths under which a landmark can be seen from different places is small and all are within the same ratio of vicinity.

The ratio between the number of active neurons in the **PrPh** matrix and the number of neurons that are really used by EC-DG is globally $\frac{N_L * \bar{n}_a}{N_L * N_A} = \frac{\bar{n}_a}{N_A}$. In order to get a good azimuth precision, N_A has to be high enough and \bar{n}_a can be small as the same landmark does not need to be encoded for too close azimuths thanks to generalization.

Therefore, it is not necessary for the **PrPh** matrix to have more columns than the maximum number of different azimuths under which a landmark can be learned. Thus, the total number of columns can be correlated to \bar{n}_a (for instance, $2 * \bar{n}_a$) and the number of rows remains equivalent to the number of neurons in the **Pr** group to which they are linked. However, in order to avoid any loss of place field in the azimuthal precision, each neuron of the matrix is linked to subset of neurons in the azimuth **Ph** group. Consequently, the neurons in the neighbourhood of the neuron encoding the position of the current landmark are all linked to the same unitary position and thus they all encode the same neuron in the **PrPh** matrix. In this way, the same landmark will not be encoded on different azimuths unless these azimuths are significantly different. How far the current landmark position is from the position of the learned landmark is immediately given by the activity value computed in the **Ph** group and directly transmitted to the activity of the corresponding neuron in the **PrPh** matrix. The further the current landmark is from the learned position, the smaller the result value will be.

At the beginning, all connection weights are set to 0. Each landmark perceived in the panorama is related to a **Pr** and a **Ph** neuron, which information is merged in the **PrPh** matrix. Therefore, when a couple landmark-azimuth is activated, the corresponding **PrPh** neuron is recruited and it triggers the learning of the corresponding synaptic connections. Thus, learning is performed on the weights between the **Ph** neuron and its associated neuron in the **PrPh** matrix as follows as well as the **Pr** neuron with the same **PrPh** neuron:

$$\Delta w_{ik}^{Pr-PrPh} = 1 \quad (2.5)$$

$$\Delta w_{jk}^{Ph-PrPh} = 1 \quad (2.6)$$

$w_{ik}^{Pr-PrPh}$ and $w_{jk}^{Ph-PrPh}$ are the connection weights between the i_{th} landmark and the j_{th} neuron azimuth respectively to the k_{th} **PrPh** neuron.

The **PrPh** neuron activity results from the product of both inputs and it is calculated by the following equation:

$$I_{ij} = \max_i(Pr_{ai} * w_{ik}^{Pr-PrPh}) * \max_j(Ph_{aj} * w_{jk}^{Ph-PrPh}) \quad (2.7)$$

Where, Pr_{ai} and (Ph_{aj}) are the activities values previously computed and correspond to the maximum activities values of the of the **Ph** and **Ph** groups respectively. A *max* operator allows choosing the maximum value coming out of the product operation.

Thereafter, the time integration process is achieved by repeatedly performing the described process (equatio 2.7) and adding the activity values of each neuron to the matrix x_{kl}^{PrPh} :

$$x_{kl}^{PrPh}(t+1) = x_{kl}^{PrPh}(t) + I_{ij} \quad (2.8)$$

This activity is reset after each complete exploration of all landmarks of a place (panoramic image).

Place cell learning Finally, the recognition of a place (panoramic image) is performed by a global correlation measure between the learned panoramic images and the current one.

The place cells group comprises of a number of neurons encoding different locations in the environment. Each different location or place is characterized by a unique landmark constellation formed by all landmarks perceived within the panorama view (process described above).

The whole group of neurons is connected to all the neurons from the **PrPh** matrix and their initial synaptic weights values are set to zero. When learning a new place, a neuron from the **PC** group is recruited. Then, for each neurons in the **PrPh** matrix, if its activity happens to be superior to 1, its synaptic weights connecting the said **PrPh** neuron to the recruited **PC**neuron is set to one. This can be summarize by the following equation 2.9

$$\Delta w_{ik,p}^{prph-pc} = \begin{pmatrix} 1 & \text{if } X_{ix} > 0 \\ 0 & \text{otherwise} \end{pmatrix} \quad (2.9)$$

Then, the activity of the P_{th} neuron of the **PC** group is calculated as follows:

$$x_p^{pc} = \frac{1}{w_p} \left(\sum_{i,k=0}^{M-1,N-1} x_{ik}^{prph}(t) * w_{ik,p}^{prph-pc} \right) \quad (2.10)$$

With $w_p = \sum_{i,k=0}^{M-1,N-1} w_{ik,p}^{prph-pc}$

Where M, N are the rows and columns dimensions of the matrix.

The resulting activity reaches the maximum value (1) when the robot is at the exact same location where it has learned the place before, and it decays exponentially with respect to the distance of the robot's current location to the learned one. This *a priori* generalization property allows the system to still activate a place cell when the robot is within the vicinity of the learned location.

2.2.5 Vision in robot navigation

An important aspect of autonomous navigation is the perception of the environment, since it provides the input for a successful control. Perception is the process of interpreting and transforming the sensory information of the state of the robot, the environment and other external entities into a representation that can be used for further processing or further actions.

This information can be extracted from different sensors of the robot, which can be classified according to what they measure into *proprioceptive* and *exteroceptive sensors*.

For instance, the information of the internal state of the robot such as orientation, velocity, position, etc., is calculated by *proprioceptive sensors* (e.g. encoder, gyroscope, accelerometer), whereas the information about the external environment such as the distance to an object, the interaction forces and so forth is calculated by *exteroceptive sensors* (e.g. laser, sonar, camera, ultrasound).

Additionally, depending on how they measure such information, sensors can also be classified as *active* or *passive*. *Active sensors* measure properties based on the response of the signal they emit into the environment such as the laser range finder, which is the most common sensor used on mobile robots. In this matter, *active sensors* exert some control over the measured signal, which makes them more robust than *passive sensors*. These latter are more sensitive to changes in the environment as they only gather data from the environment without modifying it such as cameras.

In the last three decades, navigation based on visual perception systems (such as on board camera systems) has been especially prevalent. Although the uncertainty on the measure of distance from a camera is superior to that obtained by distance sensors, cameras have the capacity to provide a perception of the environment in a single shot as well as to supply detailed information about the environment, which may be overlooked by other types of sensors. Moreover, they are lighter, less expensive and have lower power consumption compared to other sensors that are used for navigation such as infrared sensors, sonar sensors, laser range finders, position-sensing devices (PSD) and inertial sensors.

For instance, infrared sensors have limited range and are sensitive to light interference. Their reflectance is strongly dependent on the target surface, which makes them unstable when measuring distances [Benet 2002]. Even though the sonar is not expensive and data can be easily collected [Tardós 2002], [Ribas 2008], it suffers problems of big dispersal and low angular resolution.

The laser range finder and the radar supply instead a better resolution, which allows acquisition of very dense information of the environment structure [Nüchter 2007], [Thrun 2006]. However, they are more expensive and have difficulty at distinguishing various types of surfaces as well as at detecting small or flat objects on the ground. Moreover, due to their heavy

composition (large pieces of equipment), their use can be limited for aerial and humanoid robots.

Inertial navigation sensors such as accelerometers and gyroscopes provide an incremental estimate measurement of the orientation and trajectory of the moving robot. However, due to their inherent noise, the errors are cumulative and therefore such estimation is not always accurate. Additionally, they provide no information about the obstacles in the environment that the robot might encounter.

Global positioning systems (GPS) are very accurate and robust when it concerns global navigation in outdoors environments [Panzieri 2002], but they may suffer from outages due to tall buildings or forests that could block the signal, multipath effects, interference or jamming. These kind of problems usually happen in urban canyons, tunnels and indoor environments as radio signals cannot penetrate solid walls.

Due to the limited range capability and noisy-prone problems presented in almost all sensors, some researches have proposed the fusion of information from multiple sensors as a solution to more accurate and robust environment information extraction and robot position estimation [Castellanos 2001], [Sarkar 2005], [Nützi 2011]. Nonetheless, such fusion implies more power requirements and may highly increase the cost and the weight of the system.

The information about the surrounding environment that all these sensors can acquire is relatively less than what a camera can potentially obtain (e.g. color, texture and depth). For instance, small objects and different kind of surfaces can be easily detectable with a camera, which is not usually the case with range-based sensors. Furthermore, with the increasing development of lower priced processors, vision processing is becoming an affordable task that can be performed in real-time and complex applications. Therefore, many vision-based systems have been the focus of recent research work for robot navigation.

2.2.5.1 Vision sensors

Most visual navigation techniques proposed in the literature use different vision-based sensors to obtain the environmental information. Traditional cameras have a limited field of view but it is possible to increase it by using more than one camera or by placing the camera on to a rotating machinery, although this requires movable parts and accurate positioning. The biggest disadvantage of these systems is the time required to obtain a wide view, which limits their use in real-time applications.

Many systems are based on monocular systems. Even though the environment scale cannot be determined from a single camera, monocular systems have the advantage of providing rich information while being cheap, small, light and low energy-consuming. Furthermore, they can be used in either small or big open spaces, as their visual range is not intrinsically limited. [Engel 2014] presents an approach which uses a monocular camera as the main sensor to navigate a quadcopter and in order to overcome the scale-obtaining problem they combine the vision sensor with an air pressure sensor.

Other works such that of [Royer 2007] shows that outdoors autonomous navigation can be possible with the use of a single camera and natural landmarks by presenting a real-time localization system. With this in mind, they showed that the use of a camera in dense urban areas could overcome the localization problem when some satellites are masked and the GPS

localization accuracy is too poor. [Wang 2012] developed an algorithm allowing to detect moving objects while the robot navigates the environment by using a hand-held monocular camera. The algorithm is based on the epipolar constraint that is used to distinguish the moving objects from the stationary landmarks in dynamic environments. In fact, it allows to know if a set of features points satisfy the epipolar constraint on image plane given an estimated matrix. However, monocular systems are penalized by the lack of other information such as depth information. Indeed, since it is not possible to distinguish near objects from far objects, the risk of collision during navigation can be quite high.

Therefore, in order to obtain richer information, some systems use two cameras instead of one. This is the case of binocular systems, which are also known as stereo vision systems. Both cameras are placed at a certain distance from each other allowing to acquire images from different angles in order to determine the dimensions, shapes and positions of the objects as well as the depth of the image. In [Engel 2015] for instance, the depth is estimated at the high contrast pixels, including corners, edges and high textures areas in order to reconstruct a semi-dense depth map online. Harms [Harms 2015] combines a stereo camera and an inertial measurement unit in order to detect ascending stairs by estimating concave and convex line segments from depth data directly, tracking the line segments over time and fitting a stair model into the tracked line segments. There is, however, a fundamental drawback known as the correspondence problem, which consists in ascertaining if the observed object perceived in one image correspond to the same object in another image. To this end, several stereo correspondence algorithms have been proposed in the literature and [Cabezas 2012] has conducted a quantitative evaluation methodology of disparity maps, which performs an exhaustive assessment of the entire set of algorithms.

Another widely used technique is the omni-directional vision, which is rather popular among researches because of its advantages [Maohai 2013], [Valiente 2015], [Abadi 2015]. It consists of the use of a camera with a 360-degree field of view in the horizontal plane and it is usually mounted on the top of the robot to take a visual field that can cover the entire sphere. With the use of omnidirectional vision sensors, it is easier to find and track features because they remain longer in the field of view as they provide a full visibility of the surrounding environment in a single frame. However, they have a lower resolution than standard images. An Omnidirectional camera can be obtained by combining a standard camera and a convex shape mirror, such as hyperbolic, parabolic or sphere mirrors. It is also possible to combine two omnidirectional cameras in order to obtain 3D coordinates and thus form an omnistereo vision system. For instance, the system developed in [Zhang 2012], uses a hyperbolic mirror and projector to form an omnidirectional projector. It is aligned vertically with the omnidirectional camera so as to cloud-sample dense 3D points via triangulation, between correspondence pixels around all 360 degree surrounding.

PTZ (Pan-Tilt-Zoom) vision systems instead, use cameras that are usually controlled by the user with a joystick or other devices. They can be tilt up and down, pan to the left and right and zoom in and out and can overcome the limited field of view of standard cameras. They have the advantage of allowing the operator to search an entire visual area surrounding the robot without moving the robot, to track objects and to zoom closer to capture some information that cannot be seen at a certain distance or angle from the robot position. The freedom of movement that this kind of systems have, allows a better navigation; principally

in unstable environments such as urban search and rescue where a given rotation of the robot may not be allowed or where a lot of movements can cause structural damage. Nevertheless, since such cameras only records where they are pointing at, some vital information can be missed if it happens outside of their field of view. Additionally, the operator situation awareness may be affected when the camera is off-center and the operator is navigating the robot [Nielsen 2005].

2.3 World Representation

In order to achieve a successful performance in applications of some mobile robot systems requiring the execution of crucial tasks such as a safe and rapid navigation, an accurate description of the environment can be necessary and sometimes even vital.

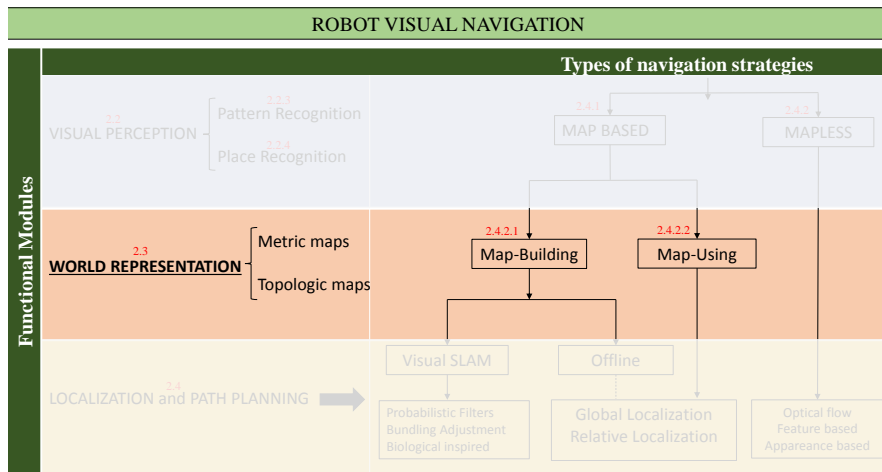


Figure 2.11: World representation. Transversal structure representing the implication of different functional modules in all types of navigation strategies.

Such an environmental description, here referred to as world representation, can be acquired from a plan of the building or designed by a human to be provided to the robot before the navigation starts (**Map-Using**). This implies that it has to be built beforehand and consequently, that the navigation environment is static and will remain unchanged. Depending on the application, this kind of maps can be sufficient for the required task.

However, this is not usually the case for most of the robotic scenarios where dynamic obstacles are prone to appear. Therefore, in other cases the world representation needs to be constructed directly by the robot through its sensors (**Map-Building**). Hence, the robot is able to consider all changes presented in the environment while navigating it.

The degree of detail of the representation can vary depending on the given task or on the application. For instance, they can go from a simple graph of interconnected silent features or objects in the environment to more complex models, such as CAD models.

World representation can be distinguished mainly in two types: *metric* and *topological maps* (figure 2.12).

Metric maps are composed of information such as distances or map cell sizes with respect to a predefined coordinate system and can be used for a more precise localization and obstacle avoidance. However, planning in a large metric map quickly grows unwieldy, as they are more sensible to sensors. Moreover, it is difficult to maintain a global consistency when closing large loops.

Among metrics maps, the most common representations are the grid-based maps and the feature-based maps. While **Feature-based maps** represent the environment by a collection of landmark locations [de la Puente 2014], [Rosen 2016], [Erinc 2014]; **grid-based maps** represent the environment in a tessellated way where each evenly spaced cell composing it, represents an obstacle (feature) or a free-space at the same location in the real environment [Elfes 2013], [Meyer-Delius 2012], [Joubert 2015], [Jessup 2014].

Topological maps instead, do not use any reference system or absolute distance among the objects it represents [Li 2015], [Johnson 2012], [Ramaithitima 2016], [Garcia-Fidalgo 2015]. It consist in building a representation of the relationship of the most characteristic features or areas in the environment based on graphs. Each feature, place, object or area in the environment is represented by a node, which is itself, connected by edges or links to the other nodes according to their topological proximity. Therefore, they can easily be used for large environments. By its simplicity and compactness, topological maps take up less computer memory, and consequently speed up computational navigation processes.

Recently, some works such as those found in [Zhang 2015], [Qin 2013], [Siagian 2014], present a hybrid map configuration by usign a combination of both metric and topological maps.

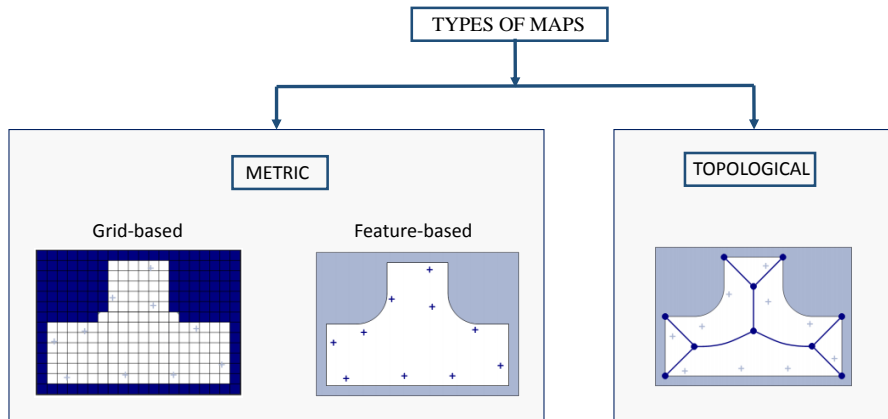


Figure 2.12: Types of maps. Grid-based: collection of discretized obstacle/free-space pixels. Feature-based: collection of landmark locations and correlated uncertainty. Topological: collection of nodes and their interconnections. (Images extracted from a work presentation of [Choset 2005]).

Although metric maps have been more widely studied for robot navigation, several works have shown the advantages of the use of topological maps. Moreover, other researches have

shown accurate results when using a variant of these types of maps with less degree of information such as floor plans [Ito 2014], [Schulz 2015], [Fallon 2013], appearance-based maps [Erinc 2014], street maps [Hentschel 2010], [Floros 2013], hand drawn maps [Boniardi 2015], [Behzadian 2015], [Yun 2008], [Kawamura 2002].

In most cases, the representation of the environment is given by a number of landmarks expected to be found in the environment during navigation, the location and identity of which is stored in the robot memory. We understand by landmark, a region in the real world described by its 3D position and its appearance [Frintrop 2008] and we consider the same distinction presented by [Fuentes-Pacheco 2015] where a salient feature is also a region but of a given image which is described by its appearance and its 2D position. Hence, while navigating the environment, the robot searches to match the landmarks that it perceives through the camera sensor to those already stored in its database. Thereafter, it searches to relate the camera sensor's measurements of the perceived landmark with the measurements of the matched landmark to find the adequate localization. This is known as the data association problem and is considered one of the hardest problems in the navigation task [Neira 2001]. When the identity of the landmarks is unknown, the algorithms provide special mechanisms for estimating the correspondence of measured features to previously observed landmarks in the map. Landmarks may be matched incorrectly, as they look different from different viewpoints. Therefore, this problem also involves determining if the measurements are spurious or belong to elements not contained in the map (gross outliers). Errors in an incorrect image matching or data association will rapidly lead to incorrect maps. Therefore, it is essential to solve this problem for a successful navigation.

Regardless the type of representation, it should be adaptable for the required task while taking into account the uncertainty inherent to both sensor data and to the robot's state estimation system. Moreover, they should be compact enough so that other components such as the path planners can make use of it.

2.4 Localization and Path planning

2.4.1 Introduction

The *localization* of the robot denotes its capacity to establish its own position and orientation in the environment. Then, once these both are established, *planning a path* leading the robot from a starting point to a final position becomes an easy task.

Generally, the position and orientation are computed according to the reference system given by a constructed spatial representation (*map-using*).

However, when the spatial representation is built as the robot navigates the environment (*map-building*), solving the localization problem is a more complex task. In fact, the use of an accurate map is quite important for the robot to localize itself; however, knowing where the robot is within the environment is also essential for building the map. This kind of chicken-and-egg problem is known as the simultaneous localization and mapping (SLAM) problem and different SLAM algorithms have been proposed to solve it.

In the absence of a world representation (*mapless*), the estimation of the robot position and orientation can be obtained directly from the use and/or combination of proprioceptive

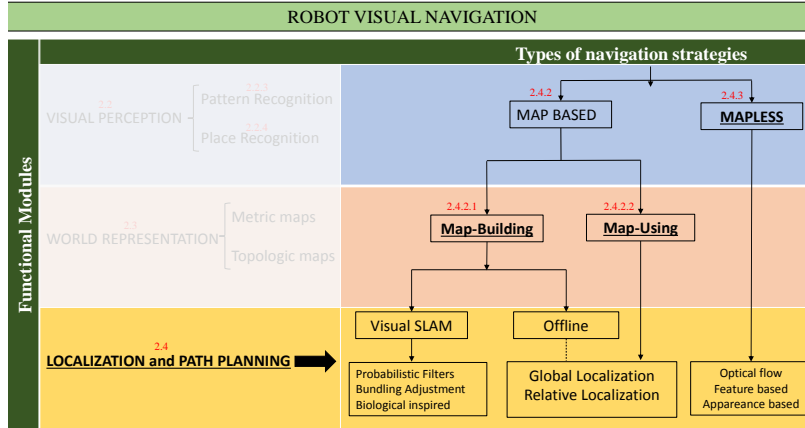


Figure 2.13: Localization and Path planning. Transversal structure representing the implication of different functional modules in all types of navigation strategies.

and exteroceptive sensors (refer to section 2.2.5 for an deeper insight of such sensors). On one hand, exteroceptive sensors such as the GPS are very robust for localization in large terrains or outdoor environments as they provide more accurate coordinates than other sensors. However, they drastically lose accuracy in indoor environments or when the terrain shrinks. On the other hand, proprioceptive sensors such as gyroscopes, allowing obtaining an estimation of the robot's position by means of a deadreckoning navigation method, inevitably diverge from the truth, as they are sensible to error-accumulation due to their inherent noise.

Path planning can be divided according to two categories based on the availability or absence of a complete representation:

Off-line path planning is performed when a representation of the world is given in advance. It is also known as the global path planning because of the use of the global information provided by the world representation. Therefore, an accurate representation of the world is essential for a successful planning.

In contrast, on-line path planning is performed when no representation of the world is given. In this case, the robot obtains the information of the environment through sensors and plan its path locally according to what it encounters (by using some visual clues as reference points for instance) while navigating the environment. A good robot localization as well as an accurate obstacle detection is thus crucial.

2.4.2 Map-based navigation

Map-based navigation techniques consist in providing the robot a model of the environment (*map-using*) so that it can navigate within the environment according to the given information or in constructing a representation of the environment (*map-building*) through the robot sensors as it navigates the environment.

2.4.2.1 Map-using navigation

The use of an *a priori* knowledge of the environment facilitates the navigation process described in the introduction above. Hence, the robot can *localize* itself in the environment by estimating its current position with respect to the recognized landmark's position from the database and *plan* a trajectory path according to the obtained information to *finally* execute its movements to achieve its final destination. That is why it is called map-using navigation.

The localization problem consists in determining the robot position in the environment. When using metric maps, the robot position is determined in coordinates with respect to a coordinate system, whereas when using topological maps, the robot position is defined to be at some place by considering its corresponding node in the topological graph. Regardless the type of map, the localization problem can be tackled from two different points of view depending on the knowledge or absence of the initial robot position: *local or relative localization* and *global or absolute localization* respectively. Once the localization task is solved, planning a path becomes an easy task. Although, it can vary according to the application or desired task. For instance, one application might need to find the shortest path because of time constraints, whereas another application might need to plan a path where the robot navigates through all places possible in the environment. Topologically, the problem of path planning is related to the shortest path problem of finding a route between two nodes in a graph.

Local or Relative localization techniques know the approximate initial robot position. Then it is necessary to update the estimate position during navigation. The simplest way to achieve this is by using an odometer. However, since the accuracy of odometry usually decreases over time as errors accumulate, external sensors are used to compensate and thus to update the new pose. More than a decade ago, [DeSouza 2002] introduced in their survey two completely different approaches to local incremental localization.

On one hand, the FINALE system [Kosaka 1992] used a geometrical representation of the space and a statistical model of uncertainty in the location of the robot. Whereas, on the other hand the NEURO-NAV system [Meng 1993a], [Meng 1993b] utilized a topological representation of the environment composed of nodes and lines graphs that represented the most representative places of the environment (central corridor, door, corners, etc.). NEURO-NAV has two main modules built up with neural networks: a hallway follower module and a landmark detector module. These two modules compute edges, detect walls and output the proper steering commands to drive the robot at a distance of a wall or centered in a corridor.

The incremental localization is also known as pose tracking and relative localization techniques attempt to find the correspondence between the measurements from the external sensors (here the vision sensor) and the information given by the map. Looking for the correspondences with these techniques is relatively easier than using absolute localization techniques, as it is not necessary to consider the entire environment but rather a small region around the estimated pose. For instance, [Biswas 2012] tackles the correspondence problem by assigning correspondences of each point from a depth camera image to lines in a 2D map. The 2D map represents the environment as a set of line segments (corresponding to the obsta-

cles in the environment) in a «vector» map and each of the points are the result of a volume reduction process of the 3D point cloud that uses the Fast Sampling Plane Filtering (FSPF) algorithm. Hence, the points are sampled and classified by local set of points as belonging to planes in 3D (the «plane filtered» points) or points that do not correspond to planes within a specified error margin (the «outlier» points). Then the localization algorithm, which is, based on an observation model down-projects the plane filtered points on to 2D which can then be compared to the lines in the 2D map.

Likewise, [Irie 2015] present an approach that copes with the problem of matching the sensor data with a street map by maximizing the statistical dependence between them. Street maps presents a more challenging effort for matching data as they lack of detailed information about the environment such as height and color. In their work, they employ a computationally efficient estimator of squared-loss mutual information to estimate the 2D position and orientation of the robot during navigation.

Global or absolute localization techniques, allow instead the robot to localize itself without the prior knowledge of the initial position. The robot position is determined with respect to a global reference frame; for instance using beacons or landmarks. Therefore, they can easily recover from positioning errors, which make them more robust than relative methods. For instance, they can handle the kidnapped robot problem, in which the robot is taken from its current environment to another unknown environment without giving it any information about the motion. Several approaches have been proposed for this purpose in the past such as Markov localization [Fox 1999] and multiple hypothesis tracking (MHT) [Jensfelt 2001]. Likewise, the probabilistic Monte Carlo Localization (MCL) method has been widely used [Thrun 2005].

The MCL uses a particle filter to estimate the position of the robot. However, even though it is robust with respect to sensor noise, it fails to estimate the robot's position if an object is blocking the sensor range of the robot. Moreover, in environments with large ambiguities, the particle filter shows slow convergence and it becomes necessary to explore a large part of the environment to solve the ambiguities and thus converge. This can be a disadvantage when one of the main requirements of a given application is to find the robot location in the quickest possible way.

Therefore, to overcome this problem [Ito 2014] present a hybrid approach based on Wifi and RGB-D data to estimate the global position. The Monte Carlo localization approach remaining the core of the approach and the use of both sensors compensate the weakness of each of them and allow an accurate and fast global localization.

Following the same line of merging the information of other sensors with that of the map, [Alonso 2012] present a global positioning solution in real complex environments that fusion the trajectory information from visual odometry with digital road maps. The motion trajectory of the vehicle is estimated using weighted nonlinear least squares (WNLS) optimization and a Gaussian multivariate model to estimate the uncertainties in the measurements [Sotelo 2007], [Parra 2010]. The outlier removal is performed by a RANSAC algorithm, based on Mahalanobis distance to better deal with the nature of the input data. This fusion allows removing the cumulative error and thus estimating accurately the position of the vehicle for very long GPS outages.

Other approaches allow effective global localization by using object recognition [Anati 2012],

[Atanasov 2014]. For instance, the Global Localization by Soft 3D Object Recognition (GLSOR-3D) described in [Ribeiro 2015] uses a soft 3D object recognition to estimate the pose of the robot with respect to the landmarks in the given map. By learning off-line the appearance of different view angles of the objects, it suffices to partially view any of the observed objects to estimate the robot pose and orientation relative to the objects. GLSOR-3D achieves this by exploiting the PVHK descriptor and the Modified Hausdorff distance as tools to recognize and compare the similarity between objects represented by their partial views.

2.4.2.2 Map-building navigation

The construction of maps refers to the process of the creation of geometrically or topologically coherent 2D or 3D models of the environment by using the robot sensors while navigating the environment. Two different approaches can be distinguished: The systems that first build a map and then use it for robot localization (offline map-building), and the systems that build the map online and simultaneously localize themselves in the environment (SLAM).

Early approaches in robot navigation considered mapping and localization as two different tasks that were treated separately. Hence, the former standard approaches divided the navigation task in two phases. In the first phase (training phase), the robot acquires some information of the environment by exploring it and builds a representation of its surrounding. Thereafter, in the second phase (operational phase), as the robot navigates the same environment, it matches the current perceived information to what it was stored before in order to localize itself in the environment. The localization task is assumed to be computed once the map has been accomplished. For this matter, most of the localization techniques referred to the map-using navigation can also be applicable in this case i.e. pose tracking, for instance by using particle filter based Monte Carlo localization.

However, after realizing that the combined mapping and localization problem was convergent, some researchers suggested that treating it as a unique problem would provide means to make a robot truly autonomous [Chatila 1985], [Leonard 1991], [Rencken 1993], [Durrant-Whyte 1996] based on [Durrant-Whyte 1988], [Smith 1986]. This meant that the robot should be able to navigate the environment while localizing itself as it navigates. The technique allowing this type of navigation is called SLAM (Simultaneous Localization and mapping), also referred as CML (Concurrent Mapping and Localization) by some researches [Newman 2002] and [Andrade-Cetto 2002]. While most proposed SLAM methods are rather robust for mapping static, structured and of limited size environments, it is still an enormous challenge to map unstructured, dynamic and large-scaled environments.

Recently, some researchers have proposed a combination of mapping algorithms with SLAM approaches in order to overcome the problems related to semi-static or dynamic environments where the location of obstacles change over time. This is called lifelong SLAM and [Einhorn 2015] propose a system that allows lifelong mapping and localization in real world applications. They combine normal distribution transform (NDT) and occupancy mapping. Hence, the map is created prior operation and it allows an accurate localization. Thereafter, when the environment changes, the system is able to update the map with the use of a graph based SLAM algorithm. Similarly but in a more general fashion, [Frese 2010] present an overview of the SLAM problem from the perspective of using SLAM for a given application

instead of giving an overview of the investigation of the SLAM itself.

Offline map-building Most of initial works back at an early stage in robotics navigation as well as some current ones perform the construction of the map in two phases. During the «training phase», the robot explores the environment and incrementally constructs a map. Then, with the aid of the built-map, the robot is able to localize itself, compute the path leading it to its final destination and thus navigate during the «operational phase».

Some approaches build the map offline out of the registered images. Therefore, they can effort to use algorithms computationally intensive, which result in accurate and rich maps. Additionally, localization techniques can greatly benefit from 3D information especially in real-world applications where there are still some accuracy issues such as in densely furnished domestic environments. Therefore, some researchers have focused their work on approaches to recover 3D environment models and use them to estimate robot motion.

Depth cameras have proven to provide a huge amount of information about the structure of the environment and [Schmiedel 2015] takes benefit of these sensors in order to build a robust representation. To this end, they use compact and highly memory-efficient type of map known as Normal Distribution-Transform (NDT) map, and introduce a new keypoint detector and descriptor called IRON allowing to accurately aligning the 3D depth maps. The robustness of the descriptor matching and outlier detection allows the system not only to build an accurate map, but also to perform both local (pose tracking) and global localization (NDT-one-shot-localization) depending on the availability of the initial pose estimation.

Another example of map-building is that of [Meyer-Delius 2012] which present a generalized version of occupancy grids proposed by [Moravec 1985] and which is still one of the most common mapping approaches in mobile robotics. The generalization consist of modeling the state changes in the representation, instead of simply considering the state of a grid cell as static. This version is suited for changing environments. They first perform a standard offline learning approach of the environment and then the robot is able to learn from its observations about the environment changes over time. Hence, the robot is capable of adapting its representation continuously and it can be used to improve the path planning performance of the robot. Similarly, but with the help of a human guidance [Kidono 2002] proposed a system where the robot is guided around the environment in the «pre-training phase». During this phase, it gathers the important information perceived from the environment and records some images allowing it to construct a 3D map online. Once the map is built, the robot is able to compute the path leading it to its final destination and it tracks the landmarks by comparing to what it is seeing during the navigation phase to the constructed map. The robot uses a stereo camera and odometry.

Visual-based Simultaneous Localization and Mapping (VSLAM)

The SLAM problem and its formulation SLAM refers to the process in which a mobile robot seeks to acquire a spatial map of its surrounding environment and simultaneously deduce its location relative to the same map while navigating the environment. This is a challenging task, as errors in a robot's position will induce errors when the map is being con-

structed, and consequently, errors in the robot position estimate [Smith 1990]. Modeling the correlation between these errors is key to a successful SLAM algorithm, and numerous solutions have been proposed (several of the most popular are reviewed by [Durrant-Whyte 2006] and [Bailey 2006]).

Any navigation environment may possibly be comprised of landmarks, objects, surfaces, etc. Therefore, one common setting of SLAM consist in assuming that the environment is composed of point landmarks such as door posts and corners of rooms, which, when projected into a 2-D map, are characterized by two coordinate values. The environment is thus, represented by a vector of size $2N$, where N is the number of point landmarks in the environment.

The SLAM problem involves recovering a model of the environment m and the sequence of robot locations x_t from the odometry and measure z_t . Therefore, to be able to solve it, the robot needs to, on one hand, relate odometry measurements u_t to robot locations x_{t-1} and x_t and on the other hand, relate measurements z_t to the environment m and the robot location x_t .

There exist different ways to address the SLAM problem. For instance, it can be distinguished between full SLAM and online SLAM, where the former seeks to recover the entire path whereas the latter seeks to recover the present robot location. It can also be addressed according to the type of map whether is metric or topological. Another common distinction can be done by assuming the environment static or dynamic (changing over time) or at another level, by assuming whether the identity of the landmarks is known or not. This *a priori* knowledge is important to solve the data association problem, which is one of the most difficult problems in SLAM. For a detailed taxonomy of these distinctions, the reader can refer to [Thrun 2008].

In addition to computational complexity and data association problem, another particular challenge for SLAM is the known «loop closure» problem, which consist of detecting if the robot has returned to a past location where it has already been after having discovered new terrain for a while. Recognizing previously mapped locations makes it possible to increase the precision of the actual pose estimate, to address the global localization problem and to recover from a kidnapping situation. Such detection is crucial for enhancing the robustness of SLAM algorithms and thus enabling additional capabilities to mobile robots.

Vision based SLAM Visual SLAM involves the use of camera images information to tackle the SLAM problem. As omnipresence of cameras has increased in the last recent years, the interest of the research has become intensified in Visual SLAM [Strasdat 2011], [Johannsson 2013], [Kerl 2013] and Vision-based SLAM [Se 2005], [Lemaire 2007] or vSLAM [Sola 2007], which employs mainly cameras as exteroceptive sensors.

Several types of sensors are used to acquire data with statistically independent errors. The statistical independence is the compulsory condition to overcome the metric movement and the noise found in the measures. The capability to obtain range information as well as environment's appearance, color and texture, makes of the camera sensor a key element for integrating high-level tasks like detection and recognition of places. Such assets enables the robot to detect loop closure situations and by consequence to accurately position itself in limited environments. Hence, different systems that use cameras as the only exteroceptive

sensor to perceive the environment have been proposed in the literature, for instance, vision-only SLAM [Warren 2014], [Milford 2008b], and monoSLAM [Davison 2007], [Perera 2011].

However, using cameras as the only sensor is a challenging task mainly in dynamic or featureless environments as the systems can be penalized by the lack of texture in some structures, lighting changes, insufficient camera resolution, many or few salient features recognition, and erratic camera motion among other factors. This may lead to common problems, for instance, problems such as data association have been addressed in [Ahn 2006], [Burguera 2014], loop closing in [Williams 2008] and large non-Gaussian error distributions in reconstructed points depth [Montiel 2006].

Moreover, some real-time systems suffer from error accumulation over time due to the camera's incapability to measure the environment scale, introducing errors into the robot's speed and position estimates. Therefore, in order to overcome this problem and to increase the accuracy and robustness of the systems, some of them use the robot odometry or proprioceptive sensors as complement to the visual sensors to extract information. These systems are known as visual-inertial SLAM [Oleynikova 2015], [Peretroukhin 2015], [Jones 2011].

Different approaches attempting to solve the visual SLAM problem have been proposed in the literature. They can be classified in mainly three different groups: probabilistic filters; bundle adjustment (BA), and finally the biological inspired techniques.

- **Probabilistic Filters:**

The probabilistic filters are the most commonly used techniques in most SLAM systems. Some of these are the Extended Kalman Filter (EKF), the Rao-Blackwellized filter, Factored Solution to SLAM (FastSLAM), Maximum Likelihood (ML) and Expectancy Maximization (EM) [Thrun 2008].

The Extended Kalman Filter offers successful results when minimizing uncertainties on small scale environments. However, it is limited in large environment as the complexity of the EKF is quadratic with respect to the number of landmarks on the map, which makes difficult to maintain large maps. For instance, although the MonoSLAM system proposed by [Davison 2003] proves its feasibility of real-time SLAM with a single camera, it is restricted to work in narrowed and indoor spaces as it employs the EKF to estimate data. However, [Clemente 2007] present an alternative approach to the MonoSLAM system, suitable for large environments and capable of performing large loops closures. In their approach, they combine a hierarchical mapping technique and a robust data association algorithm based on Joint Compatibility Branch and Bound (JCBB), a standard technique for spurious rejection within the EKF framework.

Along the same line, [Civera 2010] combine the Random Sample Consensus (RANSAC) technique with the Extended Kalman Filter in order to perform a robust estimation from data containing outliers. The available information coming from the EKF is used in the RANSAC model thus reducing the sample size, which results in large computational savings. In their work they show that, their algorithm outperforms both in accuracy and computational cost the JCBB algorithm.

The FastSLAM method instead, uses a modified particle filter for estimating the posterior over robot paths and has been proved to be faster than existing EKF-based

SLAM algorithms [Montemerlo 2002]. Therefore, [Eade 2006] present an efficient algorithm allowing real-time mapping performance in large environment by applying a FastSLAM-type particle filter to a Single-camera SLAM and combining it to a top-down search. Moreover, they introduce a partial initialization procedure to determine the depth of new landmarks that avoids linearization errors.

[Cummins 2008] propose a rigorous probabilistic approach to image matching based on the appearance called FAB-MAP. It calculates the similarity between images from two locations based on the extracted feature descriptors allowing it to determine whether an observation comes from a previously visited location or not. These properties make of it a robust system to perform loop closure detection regardless of accumulated metric error.

Another big challenge for SLAM algorithms is to work on dynamic environments. For instance, [Tipaldi 2013] use in their system a variant of the expectation maximization (EM) algorithm to learn the parameters of the representation of the environment, which employs hidden Markov models on a dynamical occupancy grid. The use of the dynamical occupancy grid allows the system to take into account the dynamics of the environment. The probability of a grid cell is represented in the analytical part of the factorization and the information learned from the representation is employed to estimate the pose of the robot as well as the state of the environment during global localization and the occupancy. Then, the Rao-Blackwellized particle filter (RBPF) is applied and its sample part represents the robot pose. Hence, an accurate and robust localization is achieved and the map is updated around the current robot location.

Similarly, in order to minimize the effects caused by the movement of the landmarks in dynamic environments [Xiang 2015] propose a graph-based SLAM. They present a mobility-robustified function to measure how stationary a landmark is in the space. An EM-based algorithm is used in order to infer the mobility scaling and estimate the pose trajectory of a robot with respect to the mobility-robustified objective function and the resulting moving landmarks are treated as outliers.

- **Bundle Adjustment:**

In vision community, the SLAM problem is referred as the Structure from Motion (SFM) problem consisting in detecting and matching different points between successive frames of a video, estimating the camera position according to the relative movements of the points, and constructing a 3D model of the environment. Hence, the bundle adjustment method is used to perform a batch optimization of the global geometry over selected images. It adjusts iteratively the pose of the camera as well as the pose of the image points in order to obtain the minimal reprojection error (between the actual and the predicted image observations), which is expressed as the sum of squares of a large number of nonlinear, real-valued function.

Initially some the SFM algorithms were carried out off-line allowing the construction of 3D representation of the environment from small sets of images. Thereafter, in the need of estimating the motion of a moving robot in real-time and computing incrementally the

surrounding environment, some real-time algorithms based on the bundle adjustment were further proposed.

For instance, [Mouragnon 2006] introduce a system that operates in an incremental way each time a new key-frame and 3D points are added to itself by using a fast and local bundle adjustment. First, they find the position and orientation of the monocular camera in a global reference frame by using a triplet of images. Then, the pose is computed by detecting and matching features of each video frame. The output is the current position of the camera and its uncertainty resulting in a complete trajectory with the 3D coordinates of matched points.

Likewise, [Eudes 2010] propose an improved version of the SLAM problem by applying a local Bundle Adjustment (LBA) on selected keyframes of a video. Their system correct the scale drift estimation of the long monocular video sequences by using information provided by the vehicle odometer; thus, the estimated pose of a new key-frame is replaced by a correct one. Computational complexity depends on several factors, including the number of images, observed 3D points, and actual image observations.

At present, incremental BA approaches become quickly computationally expensive as more information is added (camera poses and 3D points) into the optimization. Hence, large amount of information processed in a reasonable time is the focus of several BA methods. For instance, [Indelman 2015] introduce an incremental light bundle adjustment (iLBA) optimization framework that reduces considerably computational complexity compared to standard incremental bundle adjustment (iLBA is 2-10 times faster depending on the number of image observation per frame). The method incorporates two key components to reduce computational complexity: structureless BA by reducing the number of variables and incremental smoothing using adaptive partial calculations each time a new camera is incorporated into the optimization. Recently [Strasdat 2012] stated that Bundle Adjustment optimization techniques are better than filtering techniques as they give the most accuracy per unit of computing time. The conclusion is made out of a series of Monte Carlo experiments investigating the accuracy, in terms of entropy reduction, and cost of visual SLAM of both filtering and bundle adjustment. Moreover, they suggest that in order to increase the accuracy of visual SLAM it is usually more profitable to increase the number of features than the number of frames.

- **Biologically Inspired :**

Biological models and navigation systems of bees, ants, primates and humans have been the source of inspiration of several robotic systems over the last decade. They have been extensively studied and their navigation behavior respond to many of the properties that robotic navigation systems look for as a solution.

For instance, [Milford 2004], [Milford 2008a] implemented a model capable of performing SLAM in real time on a real robot based on the hippocampal complex of rodents which they named RatSLAM. The system integrates odometric information with vision sensing by modeling the place fields in rodents. Place fields are patterns of neural activity that correspond to locations in space and are modulated by the visual stimulus

and the activity of the rodent as it moves in the environment. Experimental results show that RatSLAM can operate with ambiguous visual inputs and still can recover from path integration errors. The dynamic of the network allowing multiple hypotheses to propagate and to compete with each other, helps to strengthen the belief or one of more pose hypotheses.

An improved version of the RatSLAM was later proposed by [Glover 2010] where they fusion the probabilistic local feature based data association method of FAB-MAP [Cummins 2008] with the pose cell filtering and experience mapping of RatSLAM in order to overcome to the problem of mapping and localization at different times of the day. In fact, due to continuous changes of luminosity in outdoors applications, the appearance of a scene changes constantly and makes difficult the localization task. Hence, such fusion gives a good solution to the lifelong SLAM problem.

Along the same line of the appearance problem [Maddern 2012] present a new system named Continuous Appearance-based Trajectory SLAM (CAT-SLAM), which also uses the advantages of the appearance-based place recognition of the FAB-MAP by combining them with the spatial filtering characteristics of traditional geometric SLAM. This probabilistic approach tackles the loop closure problem by improving the reliability of the appearance represented in a continuous way instead of at discrete points along the trajectory, which traverses all previously visited locations. It uses a Rao-Blackwellised particle filter to develop loop closure hypotheses over a number of observations by modeling both the likelihood of revisiting previous locations and exploring new ones.

Other extended and modified versions of the RatSLAM systems have also been proposed in the literature. For instance, [Müller 2014] adapt the RatSLAM system, initially conceived and tested for wheeled robots, to work on humanoid robots by adjusting the given constraints. Similarly, Hippo 3D performs SLAM in 3D environments with an application to a subaquatic scenario through a ROV simulation with four degrees of freedom [Albring Guth 2013].

2.4.3 Mapless navigation

Contrary to map-based navigation, mapless navigation consists in achieving an autonomous navigation without using or creating any model of the whole environment neither prior the navigation task nor online. Hence, in order to navigate the environment and localize itself the robot needs to consider some elements said landmarks, significant enough to be easily detectable in the given environment that would serve as guides for motion such as walls, doors, desks etc. Subsequently, the robot wonders about the environment, observes the scene, extracts the most relevant features of the landmarks, stores each landmark at a given position and localize itself by matching them during navigation. Since most mapless visual methods depend mainly on the vision technique or type of clues used during navigation, they can be distinguished accordingly. However, we still highlight that most of these techniques are also used in the map-based approaches for matching the correspondences and constructing maps and that is where their difference lies on.

2.4.3.1 Optical flow

When a person moves in the environment, its whole visual scene is transformed. [Gibson 1950] called this transformation optical flow and it is due to the modification of the spatial relations between the observer (person or camera) and the objects in the environment. More precisely, optical flow is defined as variations of brightness patterns in a sequence of images given by the apparent motion of the surrounding elements of the observer when the observer himself is in movement. In an image, each pixel corresponds to the intensity value obtained by the projection of an object in 3-D space onto the image plane. Thus, when the objects move, their corresponding projections also change positions in the image plane.

Optical flow is a vector field that shows the direction and magnitude of these intensity changes from one image to another. Therefore, optical flow can be used to estimate not only the motion of the objects and the nature of their structure in the scene but also the motion of the robot relative to the objects and thus infer the current robot position and velocity.

Several efficient optical flow algorithms have emerged and been used for robot navigation purposes over the last decade or so. For instance, most of them have been used to perform behaviors involving continuous motion such as corridor centering [Zingg 2010], visual odometry and obstacle avoidance, involving different flow techniques such as Camus' correlation-based method [Camus 1997], [Lucas 1981], [Horn 1981] and [Nagel 1987].

In [Moya-Albor 2016], the authors proposed an algorithm in real time that improves, in accuracy and robustness to noise and to intensity, the optical flow constraint equation of Horn and Schunck differential approach. The technique is combined with the Hermit transformation: a biological image model that describes significant visual features in digital images. As a result, the Hermit optical flow (RT-HOF) method not only is fast enough to compute an approximate solution of displacements between images but also it allows the robot to avoid mobile obstacles in two different approaches: braking and steering when mobile obstacles are close to it.

In [Honegger 2013], the authors compute the optical flow between two successive frames by using the sum of absolute differences (SAD) block matching algorithm. First, a reference block of pixels is chosen to compute its SAD value of the current and preceding frame. Then, the resulting value is compared to the SAD values within the search area and the best matching is selected as the resulting flow value. The algorithm is performed in an open source and open hardware system based on a machine vision CMOS image sensor designed by the same authors.

Thanks to its low power, low-latency and low-cost, the CMOS sensor-microcontroller is suitable for micro aerial vehicle applications where the system was tested on and which perform great results in indoors and outdoors environments. However, since the optical sensor only provides displacements in the x and y directions, information about the angular displacement of the robot has to be determined by other method. Therefore, they use an onboard gyroscope to estimate correctly the translational velocity and an automatic exposure control that allows usage in outdoor and indoor environments.

A comprehensive investigation of exiting researches on optical-flow-based robotics navigation with an emphasis on both the sensor hardware and associated reference motion models is provided in [Chao 2014].

2.4.3.2 Feature based

In structured environments, people can self-localize by distinguishing rapidly and accurately different landmarks from the rest of the scene and then navigate while tracking them. Similarly, some computational approaches attempt to solve the robot navigation problem based on the search of certain relevant landmarks that can allow the robot to minimize uncertainty in the computed pose estimation to localize itself and find its final destination.

Feature based methods permit to segment landmarks such as doors, lines, windows as well as to extract interest point features and to encode an image description of their neighborhood appearance relevant from the rest of the image in order to use them as landmarks which each define a particular location in the environment. Since most of the techniques used in this approach such as Speeded Up Robust Features (SURF) and Scale-Invariant Feature Transform (SIFT) are invariant to translation, rotation, illumination, reduction and enlargement factors, the matching correspondence is quite robust, making them ideally suited to landmark-based navigation. However, some of these techniques are relatively computationally expensive and difficult to implement in real time on a resource-constrained robot.

Therefore, in order to reduce the computational expense while still being robust at memorizing and recognizing natural landmarks, [Anderson 2013] introduces a modified but still consistent version of the SURF algorithm, the one-dimensional SURF (1D SURF). The SURF algorithm is applied to a single row of grey-scale pixels captured at the robot's horizon, hence it considers only one dimension. Since the algorithm is used for a robot moving on a planar horizontal surface, there is no need of taking into account the rotation or vertical movement of the features provided by the classic SURF algorithm. Hence, the features in the test image are matched to their nearest features of the stored image to perform landmark recognition. The robot is then able to estimate its pose location according to the information given by the matched stored image.

The use of landmarks for navigation has appeared to be a good solution when no prior information of the environment is given. However, the quantity of landmark models in the robot's memory can increase exponentially with respect to environment size and sometimes it can be redundant and costly. Therefore, in [Sala 2006] the authors study the problem of finding the optimal size of a subset of landmarks necessary to perform a robust and reliable navigation. Two views of the most widely visible landmarks are shown to the robot during a training phase. The robot forms its database, based on these views and the position at which they were acquired and then it uses it to match the visible features during navigation to compute its position and orientation. The choice of the widely visible landmarks is done by partitioning the world into a small number of maximally sized regions such that from any position within one of the regions the same set of features is visible. As consequence, the database of features is also partitioned into a set of smaller databases, each corresponding to what the robot sees in a spatially coherent region. Hence, the total number of features (corresponding to the union of all the databases) that need to be retained for localization is much smaller than that of the single database. Therefore, even without prior knowledge of the region in which the robot is located, the search is far less costly.

2.4.3.3 Appearance based

Since efficient and consistent feature extraction and correspondence is difficult in cluttered and unstructured environments, some systems algorithms rather take into account the global appearance of the image such as the color, shape, edge, texture, etc. A variety of systems allowing an accurate robot localization and navigation has been proposed in the literature making use of different holistic approaches.

For instance, [Zhou 2003] present a method that applies a multidimensional histogram on the image in order to describe its global appearance with respect to color, edge density, gradient and texture. The extra information given by the multiple histograms other than only that of the classic color histogram, allows the system to additionally describe the spatial relationship among pixels provided that a good selection of suitable image features are given into the histogram. Subsequently, the multidimensional-histogram of the current image is compared to the multidimensional-histograms of the samples database, which each correspond to a definite location. Hence, the current location of the image is given by the location of the candidate that corresponds the best according to the matching results.

This same image-matching process has been seen in insects when returning to a goal position. It is called the *snapshot model* [Cartwright 1983] and has been the source of inspiration of several computer models. However, contrary to most of the early snapshot computational models that operate on one-dimensional images, [Vardy 2003] introduce the *visual homing model* operating on two-dimensional images. Hence, the agent is able to take and store an image of the goal position surroundings and later it uses it to match it with the current perceive images. Subsequently the disparity resulting from the matching is used to guide the agent's return. While in their work, the disparity is computed by comparing images of vectors resulting from a ring operator, [Guzel 2012] adapted the same visual homing strategy to a monocular vision based system but instead, extract the key features from the images by using the SIFT algorithm. As a result, the system estimates accurately the linear and angular velocity of the mobile vehicle with an affordable computational time.

The authors in [Gaussier 1997] developed an appearance-based approach using neural networks. Inspired by biological place cells, place neurons are created in the model and each of them define a different location. The robot, in essence, merges visual information of landmarks and their azimuths to build up a spatial representation which activity provides an internal measure of localization that serves to estimate the best movement to reach the goal. [Giovannangeli 2006b] later improve this set of place- action association by achieving sensory motor tasks in indoor and large outdoors environments. More details of this approach can be found in section 2.2.4 concerning the place recognition.

Other systems have opted instead, to use artificial landmarks whose texture can be easily recognized on the environment by performing a template matching for example.

[Fernandes 2012] estimates the robot localization by using visual odometry based on observation of fiducial landmarks that are distributed on the explored environment. The first detected and identified landmarks is defined as the reference frame for robot localization and the landmark pose estimation is refined by using the method proposed by [Schweighofer 2006], which takes into account the two local minima of the estimate error function, explicitly dealing with both to find the optimal pose estimate.

2.4.3.4 Symbolic-based

This method employs a symbolic navigation approach just as people do most of the time to locate their positions and reach their destinations in unknown environments. In fact, by tracking landmarks composed of texts and directions, people are capable of achieving a successful navigation even if they are not familiar to the place. Therefore, some algorithms attempt to go beyond the detection of simple landmarks by recognizing and extracting a semantic meaning out of the landmarks in order to give more comprehensive information about the surroundings.

For instance, [Shaikh 2013] proposed a method allowing the robot to find a path automatically by detecting and reading textual information or signs located on the landmarks. First, the landmarks are located and tracked and then the semantic information of the texts and arrows within the landmarks is extracted. The resulting information is used to guide the robot to the final destination. To this end, they implement an optical character recognition (OCR) by using the Kohonen Neural Network, which allows reducing the computational cost for real time implementation. In other scenarios such as those where robots play an important role in human-robot teams, symbolic navigation not only provides a good solution but also it is necessary for achieving a successful team operation.

In [Oh 2015], the authors suggest that in order to profoundly understand the environment, robots need to be able to reason about the given commands for a specific task from their team partners besides having a good communication interaction and other high level required skills. Hence, they have developed an intelligent architecture that combines different cognitive components allowing the robot to operate at the same level of their human counterparts. For instance, in order to cooperate with humans in complex tasks, the robot should be able to understand and execute a command like «navigate quickly to the back of the building that is behind the car». To this end, one of the main components of the proposed architecture consist of performing a semantic perception to label regions and objects in the environment. They use a decision-forest classifier that labels super pixels using SIFT [Lowe 2004], LBP [Ojala 2002] and texton features [Shotton 2009] in a coarse-to-fine segmentation hierarchy and they combine it with a 3D LADAR data to separate the labeled pixels into discrete objects with coordinates in the world relative to the robot. Outdoor navigation in urban environments results are presented and they show that their multidisciplinary approach enables the robot to carry out complex tasks in various real-life scenarios without the need of any map or prior information.

2.4.4 Summary of the types of navigation

Two types of visual navigation have been presented in this section. Their classification has been done based on the presence or absence of a global representation of the environment (a map) on which robot localization and planning is highly dependent. The first navigation strategy, **map-based navigation**, addresses the strategies using or constructing a map of the environment, thereby it is divided in two groups.

First, *map-using* strategies employ topological or geometric models of the environment before the navigation tasks begins. Since the robot knows previously the global information

of the environment, this type of navigation provide fast, robust and consistent solutions to the localization and path-planning problem. However, their highly dependence on a representation of the working environment, limits the operational capability of the robot. Moreover, using a map beforehand implies that the environment is static; therefore, it is not suitable under dynamically changing environments.

Knowing the disadvantages of the first type of navigations, the second type, **map-building**, attempts to solve the problem of navigation in terms of localization and planning by constructing a map as it navigates the environment. Contrary to the first type where a person usually provides the map to the robot, this type of navigation employs the robot's sensor as it navigates in the environment in order for the robot to create itself geometric or topological models of its world. These strategies overcome the shortcomings presented in the first group of strategies by allowing autonomous robots to navigate through dynamic environments. Two different ways of tackling this problem have been presented, whereas some algorithms build the map during the training phase and navigate during the operational phase, other algorithms allow the robot to build the map and localize itself simultaneously while navigating the environment. This is known as SLAM and it is currently the most common strategy being used. However, building a robust model of the environment is computationally time and effort consuming. Additionally, map-building navigation based on visual sensors is quite challenging compared to other algorithms based on other sensors.

The second navigation strategy, **mapless navigation**, concerns a navigation strategy where the robot does not require any explicit representation of the working environment. Basically, these strategies rely on the robot sensors to capture relevant features, landmarks or objects in the environment that could serve as reference for navigation. There are three different approaches based on computer vision techniques that allow detecting, matching and recognizing different visual cues or observation in the environment. Besides, a fourth approach involving motion estimation techniques is also widely used. The inspiration behind mapless navigation lies on the behavior found in living organisms and thereby resembles the most the human behaviors. However, the lack of a global representation of the environment can limit an optimal localization and can slow down the navigation process.

2.5 Conclusions

This chapter has presented a transversal structure describing the implication of the functional modules in each type of navigation strategies found in the state-of-the-art: mapless navigation and map-based navigation composed itself of map-using and map-building navigation. While the navigation strategies are described in terms of the localization and path planning functional modules, the visual perception and the world representation functional modules are described separately. The visual perception section highlights a biological place recognition approach our work is inspired on and the world representation section introduces shortly the two types of maps used in this work.

Robot navigation needs a control unit capable of organizing, unifying and monitoring the various components of a robotic system. Therefore, the next chapter reviews in detail the different control paradigms allowing to define the capacities of the robot to plan a line of

action and execute a variety of intelligent behaviors according to what it senses as well as to its interactions with the environment.

Control architectures

Contents

3.1	Introduction	67
3.2	Control Paradigms	68
3.2.1	Design viewpoint	69
3.2.2	Functional viewpoint	70
3.3	Conclusion	87

3.1 Introduction

In the navigation task as in other robotic tasks, the input information of a robot system either coming from the robot sensors or from an available representation of the world, needs to be processed so that the robot can make autonomous decisions on how to act on the environment and follow their execution. Such process, most of the time complex because of the application domain, requires a control system capable of monitoring and coordinating all the robot components as well as their inner interaction respectively.

According to [Mataric 1992] an architecture provides a structure for organizing a control system. They allow to structure the different levels of development into levels of abstraction as well as to improve the reusability and modularity of hardware and software components of the robotics systems. Hence, robot control architectures could be defined as control schemas developed to integrate different functionalities and capabilities endowing the robot with an autonomy to plan its line of action and produce intelligent behaviors.

A control architecture is required to meet some design properties and behavior specifications. Here below, we present some of those found in [Alami 1998], [Nakhaeinia 2011], [Brooks 1986].

Global reasoning: Reasoning in a global way gives a better insight of the past event mistakes and helps to plan ahead optimal ways of achieving a given task. A high level decision-making requires of a good understanding of the overall situation.

Reactivity: all different components of the architecture must be capable of appropriately reacting to the specific received stimuli, especially when unforeseen changes appear in the environment.

Integration: The software tools that run on the robot are very diverse. Therefore, it is important to propose good tools for every type of component, and especially to offer transparent mechanisms of communication and exchange of the data.

Robustness: The architecture has to allow using the redundancy of information sources, processing, and the multiplicity of processors. Moreover, it has to be invariant to imperfect inputs, unexpected events and sudden malfunctions.

Resolving multiple tasks: It is inevitable to find situations where conflicting concurrent actions have to be performed. Therefore, a control architecture should be able to decide on the priority of each tasks while providing means to fulfill all the multiple tasks.

Reliability: The use of robots in critical situations (for the robot and its surrounding environment) requires the use of methods, which guarantee certain safety properties. The architecture should provide the robot a good performance without failures or degradation of it.

Programmability: A highly and easily programmable machine (both from the point of view of the programmer and that of the user) would allow a robot to achieve different tasks described at some abstraction level, instead of only one precise task. From the functional level to the decision-making level, it should be possible to program control loops and low-level processing, functioning constraints and procedures of goal refinement, among many others.

Flexibility, modularity, expandability: Since the conception, implementation and building of different architectural components usually takes long time, the architecture should be flexible enough to add new features without questioning or modifying the already existing ones at whatever level they might be.

Autonomy, adaptability, coherence: The robot should be able to execute the actions, refine and adapt its plans and its behaviors according to its goals and to the environment as it perceives. Sometimes if not often, the environment changes unpredictably, therefore, the robot has to adapt to these changes. Additionally, its behavior and its reactions should be guided by its goals.

3.2 Control Paradigms

Diverse architectures of control have been proposed in the literature to design and develop strong, flexible, reliable and high performance control systems. Each of these architectures of control involves new concepts and solutions to solve the robot navigation problem based on the use or combination of different paradigms that have emerged since the early days of autonomous robot conception.

A detailed description of each paradigm is given below according to two different viewpoints. On one hand, the functional viewpoint classifies the paradigms in terms of their

internal functionality and thus in terms of their capabilities to act on the environment when performing a given task. On the other hand, the design viewpoint specifies two paradigms based on how the data information is processed and propagated through the systems as well as how the knowledge is ordered.

3.2.1 Design viewpoint

From the design point of view, control architectures can be distinguished according to two different methodologies that have traditionally been used: top-down and bottom up.

They differ in the way of how the sensory data is processed and propagated through the systems as well as how the knowledge is ordered. These two opposed strategies have their roots in the field of Artificial intelligence and have been the basis on the design of autonomous robots in the robotics field. Both approaches have their advantages and disadvantages, there where one excels, the other fails and vice-versa. This trait of both approaches is the root of a debate among robots designers on how to build autonomous robots.

3.2.1.1 Top-down

The top-down approach was the dominant paradigm in the early days of AI robotics (1960s-1970s) when researches considered that creating a machine with artificial intelligence could be possible by reducing human intelligence to symbol manipulations. At that time much of the focus was on robot planning. Therefore, in order to perform high-level tasks, a pre-programmed global knowledge was first given as input to the system. Then, the information was decomposed into smaller subdivisions and so on until each of them was reduced to basic elements that could be specified and explained by themselves. Whenever, one of the subdivisions could not be specified, it was replaced by a «black box» and thus, manipulated by the system to obtain the desired output. However, the absence of knowledge and specification of a given subdivision could not give clarity to the comprehension of elemental mechanisms and sometimes the validation of the complete model.

Since an optimal robot navigation based on this approach relies on prior knowledge information of the environment, the required amount of information to be stored in the robot's database as well as the computing complexity can be huge and difficult to handle. Moreover, this implies that the global information is not going to change in the middle or after the process. Therefore, since the information is centralized, whenever there is a modification, the whole navigation task will collapse [Clancey 1991].

Nonetheless, the systems based on this approach have a good understanding of the environment; thereby they have better capacity of reasoning when the environment is to remain static.

3.2.1.2 Bottom-up

By the mid-1980s the top-down paradigm of symbolic AI was being questioned, thereby giving popularity to the bottom-up paradigm. The robots were slow and had trouble operating in complex and dynamically changing environments because they had to plan all of their actions

based on internal world models. Hence, bottom-up models proposed an alternative to such shortcomings.

Contrary to the top-down paradigm, the bottom-up paradigm does not require of a centralized control or high-level organization of the system. Consequently, it allows reacting to unforeseen situations by quickly controlling the robot's movements without requiring complex computer programs. In bottom-up models, the design process starts with specifying requirements and capabilities of individual components. They are relatively simple processing units connected in a network that by interacting among themselves and with the environment, produce complex and more 'intelligent' behaviors. The global behavior is said to emerge from such interactions. Hence, a system based on this approach is able to build its own knowledge and learn by itself from the interaction with the environment, usually performed with parallel processing such as neural networks.

Even though, these systems are quite simple, adaptable and flexible, they lack practicality. For instance, if a robot is needed in a disaster scenario, it has to act as fast as possible. Therefore, respecting the time constraint is crucial for a successful task achievement and a robot without the previous knowledge of the navigation environment would badly fail.

3.2.1.3 Summary of the approaches

Even though the bottom-up approach is much younger, it has shown to have a value and a place in robotics as much as the top-down approach has. Whereas **top-down** models allow breaking down the problem into low-level commands helping the robot to plan its future movements, **bottom-up** models are suitable systems for navigating in unknown and dynamic environments. Their parallel processing enables the robot to learn to deal with unforeseen situations and difficulties. **bottom-up** models can easily adapt to any changes and do not require huge computing complexity as **top-down** models do.

However, they both present individual shortcomings that still need to be overcome. Due to its sequential process information, the navigation process in **top-down** models can be delayed and the huge amount of space required for storing all the preprogrammed knowledge can exceed the robot space memory. Moreover, the malfunctioning of one of the modules can cause the failure of the entire system. Likewise, with a **bottom-up** approach is very hard to achieve a higher-level complexity and the required time to learn a task or to achieve an intelligent behavior can be a limitation in time constraint tasks.

A solution instead, would be to combine both, **top-down** and **bottom-up** approaches in a possible way for a control architecture to have a preprogrammed knowledge of the environment, while being able to adapt to the real world environment thanks to emerging behaviors resulting from the interaction with the environment.

3.2.2 Functional viewpoint

Four different types of control paradigms allowing the robot to execute an action according to the perceived information can be distinguished. However, they differ from one another in their internal functioning and thus in their capabilities to act on the environment in terms of a given task.

In his book «Introduction to AI robotics» [Murphy 2000], Murphy describes and illustrates these paradigms in terms of the relationships between three primitives, sense, plan and act (SPA paradigm). Certainly a point of view with roots in the classic symbolic AI.

Sensing concerns the function of taking information from the robot's sensors and translating it into an internal world model or an output useful for other functions. *Planning* instead, is more complex as it takes the information either from the sensors or from the internal world knowledge in order to produce one or more tasks for the robot to perform. Finally, *acting* is the task of producing output commands to the robot motor actuators



Figure 3.1: Sense, Plan, Act (SPA) Paradigm.

This paradigm has its roots on the symbolic AI paradigm of the 1950's and for several years it was the dominant paradigm (deliberative) for building robots capable of imitating human intelligence, contrary to the Reactive paradigm which functioning is based on a simple stimulus-response mechanism [Wiener 1961] (see figure 3.2). They are both distinguished mainly by the speed of reaction, the consideration of the global knowledge of the world, the usage of perceived data and the computing complexity.

However, as they both presented some shortcomings, the hybrid paradigm was conceived combining advantages of both deliberative and reactive approaches while diminishing their individual drawbacks (see figure 3.3). As a result, hybrid control architectures employing a hierarchical/sequential division are composed of both components: deliberative and reactive.

Hence, while the deliberative paradigm follows the sense-plan-act process, the reactive paradigm omits the planning by only sensing and acting directly through the robot actuator (sense-act); the hybrid plans at one step and the sensing and acting are done together (plan, sense-act).

However, Brooks rejected the symbolic (deliberative and hybrid) paradigm and focused on the development of basic process unit that allows robots to move [Brooks 1986]. He considered that the capacity of reaction of a robot under unforeseen situations was an important quality to take into account under dynamically changing environments, particularly when they are unknown and uncertain. The symbolic paradigm did not allow the robot to react accurately to real time environments. Therefore, he proposed a new architecture based on a behavior-based paradigm consisting of a collection of simple processing units called behaviors connected each of them directly to the robot sensors and actuators (see figure 3.4) allowing to perform a successful navigation.

Hence, the emergence of the behavior-based control paradigm started to get the attention of few researches based on the nouvelle AI, as it was possible to enable the robot with learning capacities while still performing other tasks as good as the other paradigms did. It can be considered as an extension of the reactive paradigm but with more complex functionalities where a learning process takes place.

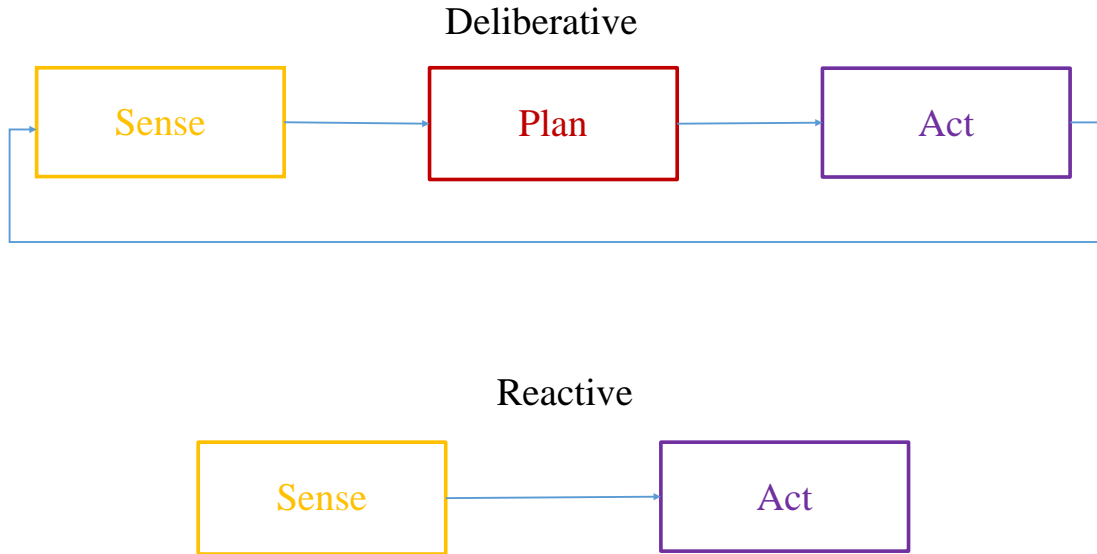


Figure 3.2: Deliberative and Reactive Control paradigms in terms of the relationships between three primitives, sense, plan and act and in terms of how sensory data is processed and propagated through the system [Murphy 2000].

3.2.2.1 Deliberative

The deliberative control approach involves the employment of reasoning techniques to decide on actions to take based on a model of the environment. It has its roots in the traditional artificial intelligence (AI) paradigm, in which a central planner fuses all sensors readings, builds a model of the environment, finds a path, plans the next action, and finally steers the robot. The robot data processing is realized under a sequential form. Therefore, the navigation process based on the aforementioned functional modules can be incarnated in a hierarchical architecture (see figure 3.5) as presented by Brooks in his work proposal of an alternative paradigm [Brooks 1986]. The robot acts on the navigation environment according to the internal environmental representation that has constructed provided by the sensory data.

This paradigm was the foundation of many robotic control architectures for many years [Schwartz 1983], [Chatila 1985], [Takahashi 1989], [Latombe 1991].

The Shakey robot was one of the first robots which architecture consisted of these three functional elements [Nilsson 1984]. The sensing module was in charge of building a map of the environment out of the images taken by the robot's camera. Then, the planning module used this map and the information of the goal destination in order to plan a path leading to it from a starting point. Then, the path, represented by a series of actions, was sent to the robot output by the executor module.

As in most of deliberative architectures, the actions taken by the Shakey robot were executed directly without needing to reuse the sensors that created the model. However,

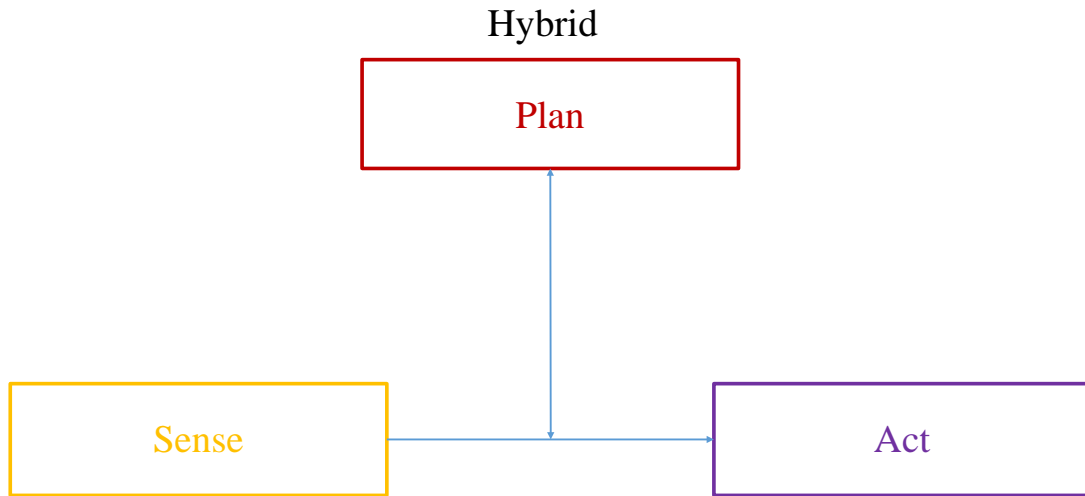


Figure 3.3: Hybrid control paradigms in terms of the relationships between three primitives, sense, plan and act and in terms of how sensory data is processed and propagated through the system [Murphy 2000].

planning required the use of an accurate model of the robot environment and building such models implied the use of high-precision sensors, which are often expensive.

Although this model allows the robot to reach a specified goal by generating optimal sequences of actions in a complex environment, unforeseen changes such as new obstacles pose a major hurdle in task completion. The need to process a complex hierarchy of information at every step can further slow down the progress through highly dynamic environments since there is a strong sequential interdependency between modules. The delay caused by one of the modules will delay the next one and so on. Therefore, since this type of architectures is limited by the lack of real time reactivity, especially in complex and dynamic environments; purely deliberative architectures are rarely used in physical like the one presented by [McGann 2008], who focused on the control of autonomous submarine vehicles.

Alternative architectures have emerged through the course of time modifying and improving the deliberative ones.

3.2.2.2 Reactive

The reactive control approach uses a stimulus-response model that defines the movements of the robot as a consequence of a stimulus. Such performance is similar to the behavior observed in living organisms like insects where navigation is exclusively based on the sensory perception and execution of simple behaviors. It can thus respond robustly and rapidly under dynamic and unstructured environments.

In order to explain such natural behaviors in an evolutionary way, [Braitenberg 1986]

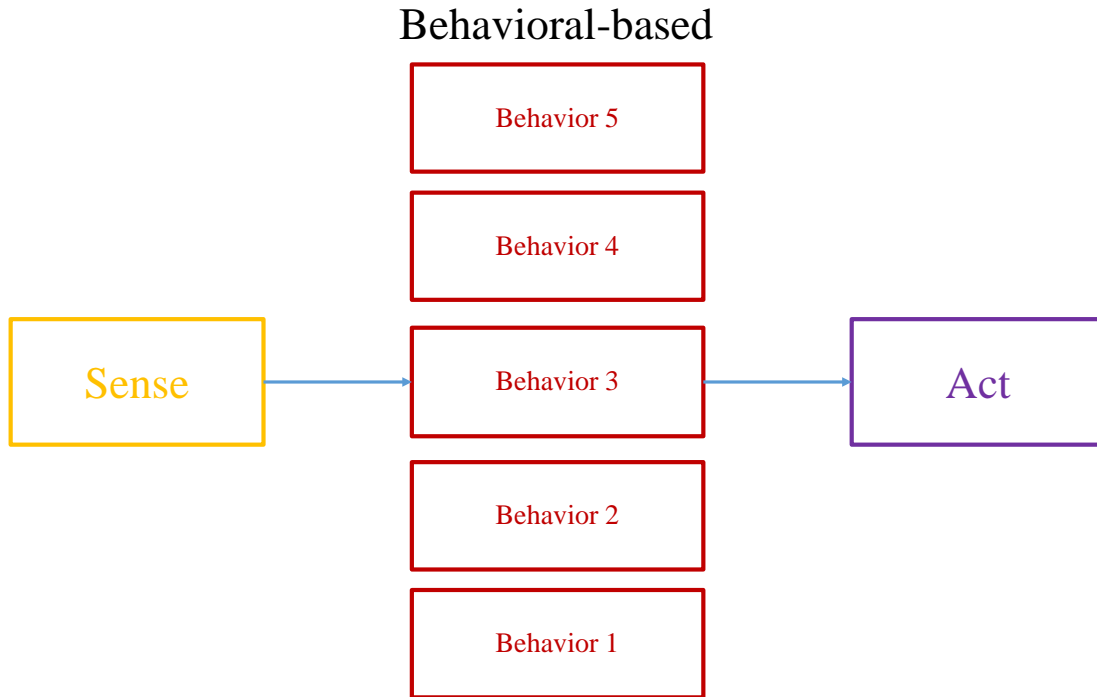


Figure 3.4: Behavioral paradigm in terms of the relationships between three primitives, sense, plan and act and in terms of how sensory data is processed and propagated through the system [Murphy 2000].

presented a concept of vehicles conceived in a though experiment. The motion of each vehicle is directly controlled by a group of primitive sensors and depending on how they are connected to the wheels (each driven by its own motor); the vehicles exhibit different behaviors that may appear complex or even intelligent. This evolutionary approach inspired many researchers that subsequently set this concept to practice.

Hence, based on this approach, from the 80s, a new generation of robots designers tried to build robots without the use of internal maps of the world. With the use of architectures based upon layers of perception-actions mechanisms, it was possible to explore alternatives to the approach based on symbolic descriptions that were stored in the robot. This perspective is called situated robotics and refers to embodied machines existing in complex and often dynamically changing environments which behavior is strongly influenced by it and the situation.

One of the first known robots examples were invented by Brooks [Brooks 1986] and one of the purposes of this approach was to develop a spatial learning theory without defining predefined categories within the architecture of the robot. More precisely, instead of storing an internal representation of the world and examine it to perform actions, the robots of Brooks reacted directly to local sensations while interacting with the environment.

Hence, reactive navigation strategies are local strategies that use actions reflexes. More

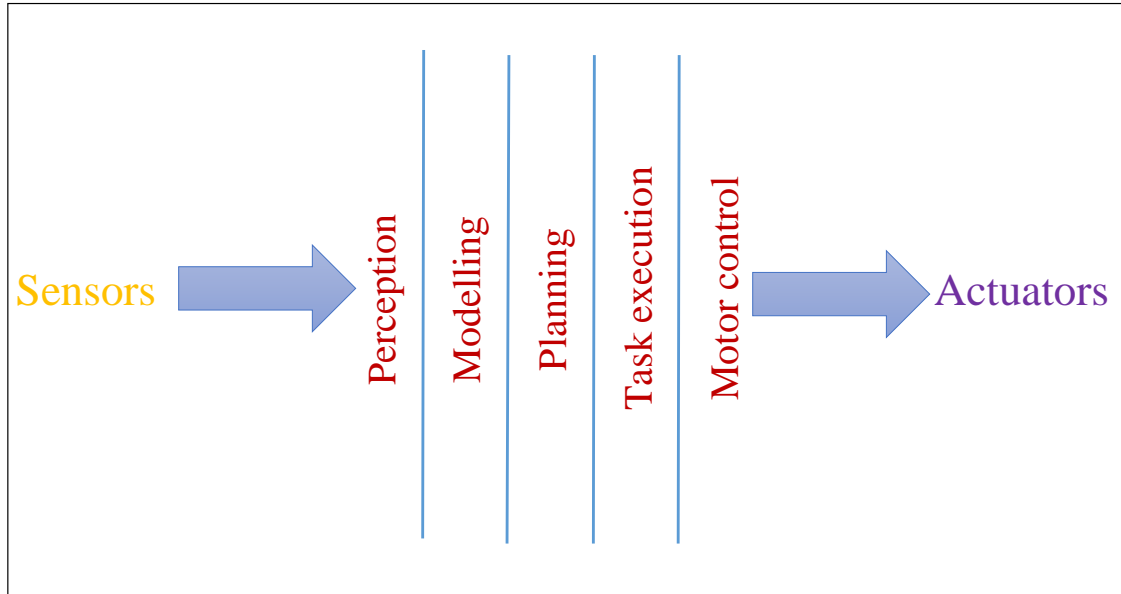


Figure 3.5: Decomposition of a mobile robot control system into functional modules by following the SPA paradigm (decomposition detailed by Brooks [Brooks 1986])

particularly, they work correctly only in the zone of their goal visibility and the actions associated with the motor are directly related to the current perception values of their sensors. Therefore, reactive architectures neither need a global model or prior knowledge of the environment nor rely on complex reasoning processes (usually utilized in deliberative architectures) to decide what actions to perform. Sensory data is distributed to individual reactive modules and the information is processed in parallel rather than sequentially. The information gathered by the sensor is the input of the set of behaviors, which transforms it into a desired response and only the most appropriate and dominant behavior finally executes the action (see figure 3.6).

Thanks to their simplicity, behaviors are executed very quickly thereby allowing to perform low-level tasks such as moving towards a given goal or avoiding unexpected obstacles in unknown environments.

However, the probability and extent of success of quickly processing the information and reacting under complex environments with these systems may be affected by the lack of an overview of the environment for reference. Moreover, the complexity of tasks the robot can address are limited notably because of the inability to have a memory or a capacity to store enough information and consequently the inability to learn and improve over time.

Braitenberg Vehicles Valentino Braitenberg [Braitenberg 1986] describes in his book the concept of a «vehicle» as an intelligent agent that can move autonomously within an environment. This concept is conceived in a thought experiment that illustrates in an evolutionary way, the capacities of simple agents. He uses nature as inspiration in order to compare these



Figure 3.6: Reactive architecture. The sensors are directly connected to the actuators

extremely simple vehicles with animals by referring to a psychological language to describe their behaviors. Hence, the evolution of these vehicles reflects the evolution of the animal species. They represent the simplest form of artificial intelligence based on behaviors or physical knowledge: for instance, the intelligent behavior that arises from the sensory-motor interaction between the agent and its environment, without the need of an internal memory, representation of the environment or inference.

The objective of these experiments was to illustrate some essential aspects of the internal structure of the animals' brain. Every experiment includes the description of a simple vehicle provided with a small group of primitive sensors, capable of measuring stimulations and connected to the motors vehicle which immediate response directly appears from a signal or a stimulus in the sensor (similarly to the neurological connections in animals).

Hence, he shows how the resultant vehicles are capable of realizing different complex behavior, according to the interconnection between the sensors and the motor, which can be described as a fear, an aggression, an attraction, a logic, etc. There exist 14 vehicles and each of them present the essential characteristics of all the vehicles, which precede it, but it adds it a stage of evolution to reach a threshold of greater complexity. For instance, vehicle 1, «alive» in figure 3.7 consists of a single sensor and a motor wheel. The latter accelerates and moves forward towards the direction at which it points, when the sensor is stimulated by a source and it slows down as it goes away from the source.

In the case of a source of heat for example, the vehicle will always try to stay near a warm place and will flee the cold. Vehicle 2 in figure 3.7 has two opposing behaviors «timid» and «aggressive», that depend on the connection of its sensors with the wheel motors. It consist of two sensors and two wheel motors. In the first case, 2a, each sensor is linked to the wheel motor of the same side and whenever one of the sensors is stimulated, its associated wheel accelerates more than the one far from the stimulus. Consequently, the vehicle goes away from the source (afraid or timid). On the contrary, in the second case, 2b, each sensor is linked to the wheel motor of the opposite side. Then, when one of the sensors is stimulated by the source, the associated wheel (from the other side) accelerates, while the other remains immobile making the robot move towards the source as it was going to attack it (aggressive).

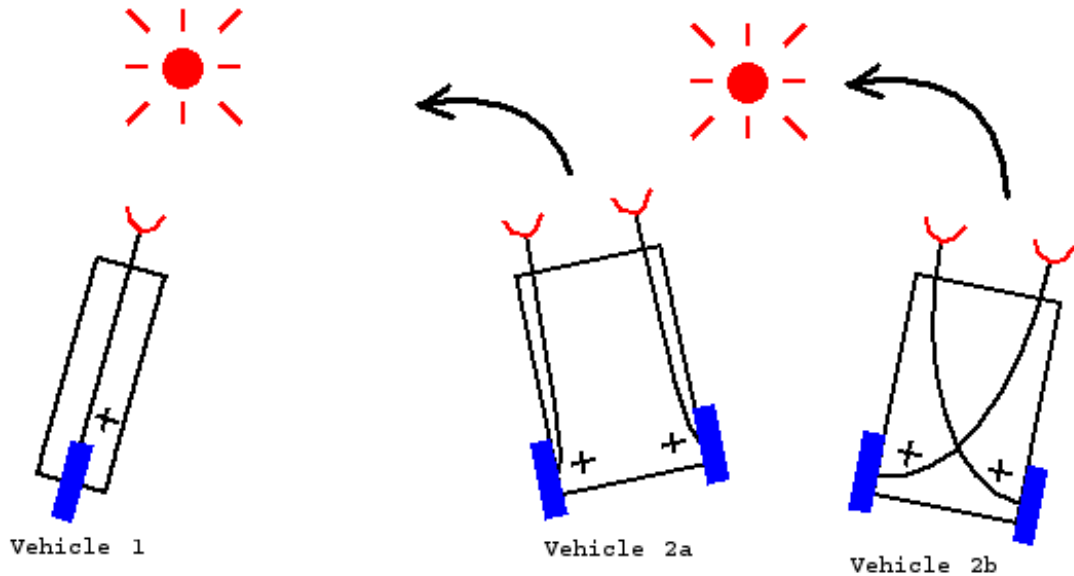


Figure 3.7: Braitenberg vehicle 1: alive. vehicle 2: afraid(2a) and aggressive (2b) vehicles [Braitenberg 1986].

3.2.2.3 Hybride

Hybrid control approach was conceived in the need of overcoming the limitations exposed by the reactive and deliberative control approaches. Hence, by merging these two approaches it is possible not only to mitigate their individual drawbacks but mainly to improve the performance of the robot navigation with the advantages of both approaches: the speed of reaction, global knowledge of the world, usage of perceived data and computing complexity. As a result, hybrid control architectures contain both components: deliberative and reactive; and they can be classified mainly in three different styles according to what has been developed so far:

Managerial Style: The architectures calculate a conventional complete path in the deliberative module that controls the behaviors and actions in the reactive modules [Arkin 1989], [Yavuz 2002]. Then, the reactive modules generate the adequate action by solving any problem that might appear. If a given module cannot solve the problem, the superior module takes over. For instance, in the SSS (Servo, Subsumption, Symbolic) architecture, a symbolic planner controls the reactive module [Connell 1992], whereas in the CoCo (Cognitive Control) architecture, the deliberative module advises the reactive module through a set of motivational variables [Qureshi 2004]. A small variation of these works can be seen in [Low 2002] where the planning module produces a sequence of checkpoints that work as sub-goals leading to the final target instead of the entire path.

State Hierarchies style: This style uses the knowledge of the robot's state in past, present and future in order to generate the robot motion [Peter Bonasso 1997], [Lindström 2000].

While the deliberative module use the knowledge of the robot's past state in order to predict the future (path planning), the reactive module functions in the present state (self-awareness) and follows the deliberative planning instructions to achieve the final action.

Model-oriented style: Alike to the deliberative architecture, this style of hybrid architecture concentrates more on the global model of the environment to navigate the environment [Konolige 1997] but it uses the reactive module to update the model and thus reduce processing time that it requires [Davies 2008].

Most common hybrid architectures are composed of three layers usually organized in parallel as illustrated in figure 3.8.

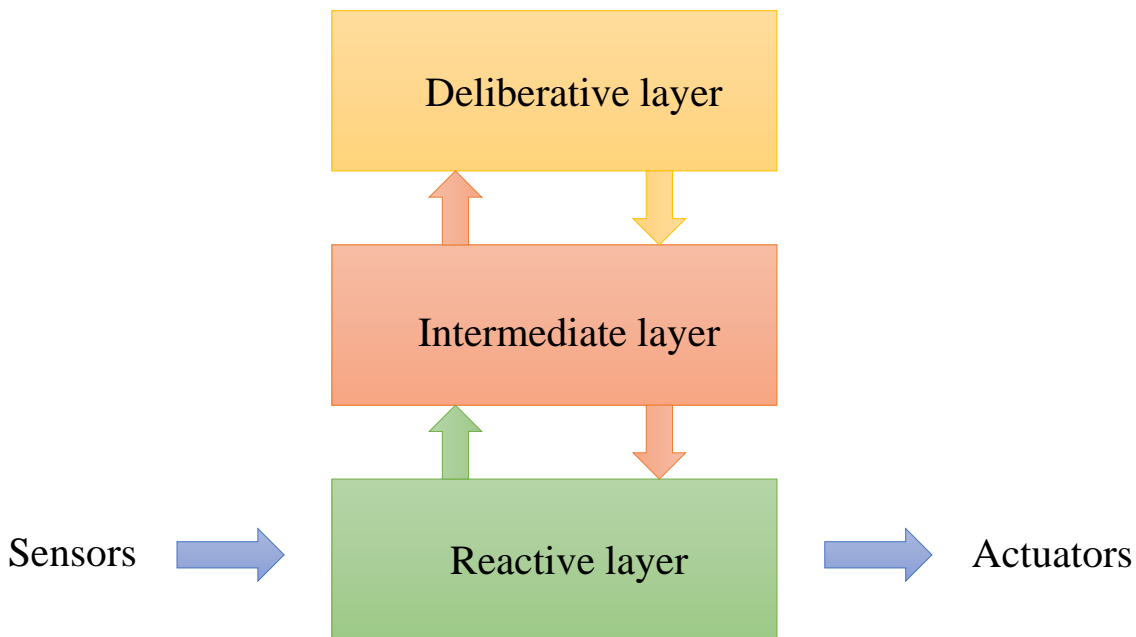


Figure 3.8: Hybrid paradigme : most common type of architecture composed of three layers

While at the bottom of the architecture the reactive component deals with the most urgent tasks as fast as possible, at the top of the architecture resides the deliberative module operating the highly abstract, symbolic and internal representations of the world for achieving long-term goals.

However, in order that the deliberative module performs a plan for high-level decision-making and the reactive module accomplishes obstacle avoidance for instance; both modules have to interact with each other to produce a coherent output accordingly. Therefore, the third layer is to be placed in the middle of both layers, serving as an intermediate component to reconcile both representations and to resolve any conflict between their outputs. It acts as the coordinator of the system and it plays an important role in the good performance of the system. For instance, as the top layer is responsible for generating an optimal plan, the coordinator layer decomposes the task into low-level subtasks (behaviors) and chooses

the most appropriate one to send it to the reactive layer to generate the robot's action. Additionally, it maintains temporal constraints between the subtasks allowing producing the best output according to the situation.

Nonetheless, interfacing these fundamentally differing components is a complex issue and the search for the optimal way to achieve a hybrid solution is still an area of ongoing research.

3.2.2.4 Behavioral

The behavioral approach or behaviorism is a psychological approach based on the proposal that any action executed by a body, including thoughts and feelings, can and must be seen as a behavior. According to behaviorism, individual responses to various environmental stimuli shape our behavior. Behaviorists believe that behaviors can be studied in a methodical and recognizable way regardless the internal mental states. Thus, any behavior can be clarified without the need to think about mental psychological states; intelligence results from the interaction among a set of asynchronous behaviors and the environment. The major principle of this approach is based on the analysis of human behavior in the stimulus-response interaction and the association between them.

Thorndike [Thorndike 1913] was the first behaviorist to explore the field of study that establishes learning as a set of associations on particular process of behaviors and the consequences thereof. This behaviorist theory of learning by stimulus-response is then developed by [Skinner 1974], as an operational conditioning which considers all learning as being a base of resulting habits of a reinforcement and a reward. Most systems are reactive, which means that they barely use the internal state to model the environment.

Behavior-based control systems do not present the constraints of lacking of a representation of the world or having little (if any) state as reactive control systems do. On the contrary, the collection of behaviors composing such systems do have states, which allows to construct representations, thus enabling reasoning, planning, and learning. Therefore, there is no need and thereby it is rare that a behavior performs an extensive computation based on a traditional representation of the world.

Behavior-based control systems were developed for situated robots allowing them to react and adapt to changing environments by simply coupling perception with action through a set of behaviors with no centralized world representation as illustrated in figure 3.9.

Behaviors are a set of distributed and interacting control modules allowing the robot to achieve and maintain a given goal by taking inputs from the sensors, generating a desirable output and sending it to the actuators. As the system is built, new behaviors can be implemented incrementally starting from the simplest and often reactive in nature survival behaviors such as *obstacle avoidance* and following to the ones providing more complex capabilities like *landmark-finding* or *homing*.

All behaviors are executed concurrently allowing speed of computation; therefore, sensors and actuators can be used independently by many or all behaviors. However, this implies that the system have to choose a particular action or behavior out of the multiple options that might appear. This process is known as the *action selection* or *behavior coordination* problem and is still one of the biggest challenges in behavior-based control systems. Specially, in applications that go beyond the navigation task as it is difficult to ensure *a priori* the

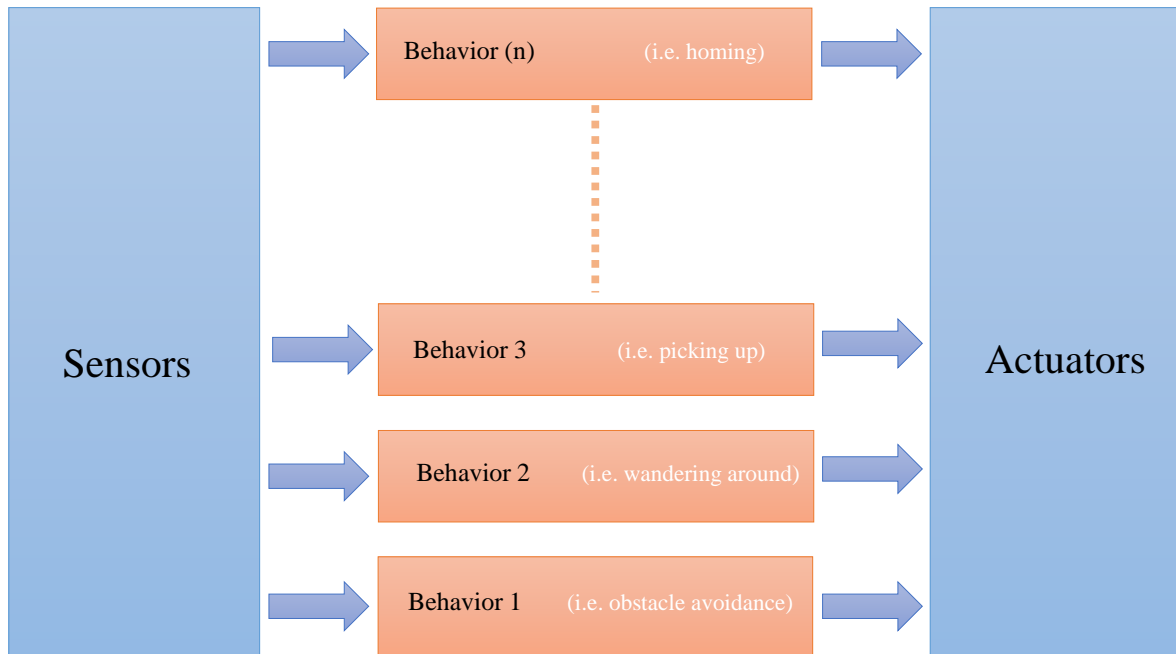


Figure 3.9: Behavioral paradigm, composed of several distributed and interacting control modules called behaviors

execution stability of the complex enforcement law such as those required for the control of robotic arms. Several approaches have been developed in order to solve this problem by providing increase flexibility; however most of them tradeoff on the efficiency or analyzability of the resulting control system.

The same functional modules presented above in a hierarchical form are achieved by a combination of different behaviors in a bottom-up approach as proposed by Brooks [Brooks 1986] (see figure 3.10).

Brooks introduced the term of the subsumption architecture, which become subsequently the best-known and most influential behavior-based architecture in autonomous robotics. For instance, the DAMN (Distributed Architecture for Mobile Navigation) architecture proposed by [Rosenblatt 1997] is another variant of Brooks' work. Concurrently, Arkin proposed another technique that came to be known as Motor-Schema architecture [Arkin 1987]. Both of these methods use behaviors to generate timely response in dynamic and unstructured real world navigation scenarios. While the output of the Subsumption architecture results from competitive selection of behaviors, the output of motor Schemas architecture results out of a co-operative coordination of behaviors. Other developed methods include varieties of motor schemas [Arkin 1989], command fusion [Payton 1992], spreading of activation through a behavior network [Maes 1989], [Maes 1990] and fuzzy logic [Saffioti 1997], [Michaud 1997] among many others. For a survey of action selection mechanisms, the reader can refer to [Pirjanian 1999].

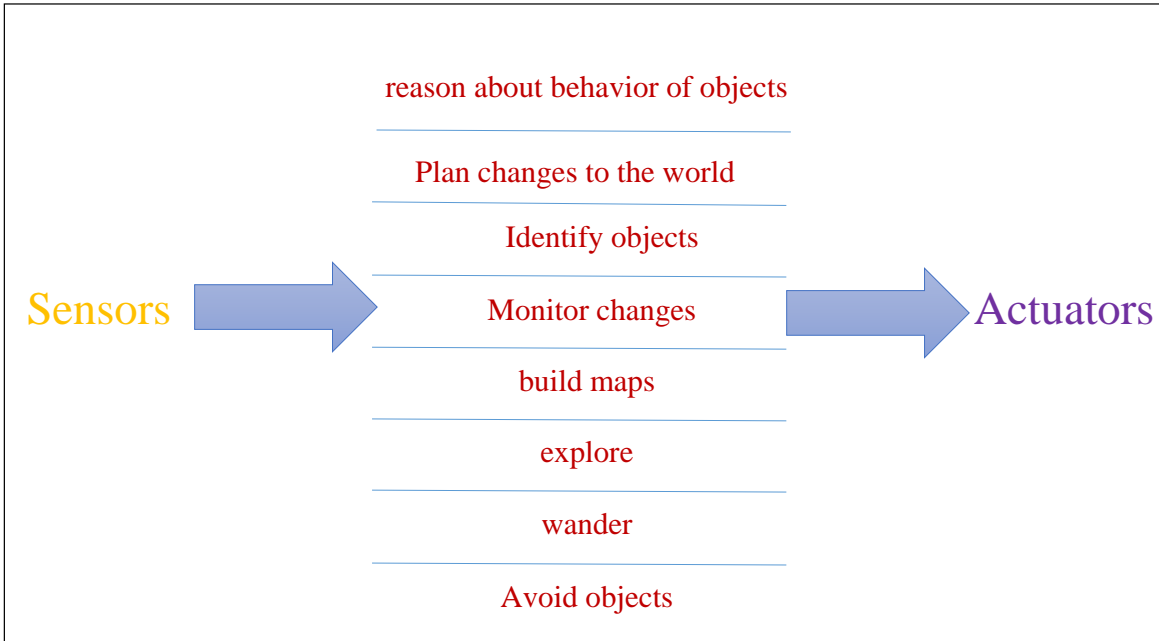


Figure 3.10: Decomposition of a mobile robot control system based on task achieving behaviors (decomposition detailed by Brooks in [Brooks 1986])

Several forms of distributed representations have been used such as a network of landmarks as in [Mataric 1991], or a network of parameterized navigations behaviors as in [Nicolescu 2001]. In this latter, a behavior is assigned to each newly discovered landmark. The descriptor information such as the coordinates, type and orientation of the landmark is stored in the behavior and whenever the sensory inputs matched the landmark descriptor to the perceived landmark, the behavior becomes active and the robot can easily localize itself. Subsequently, planning becomes an easy task since all behaviors encoding a landmark are connected to each other within the network and by using a message-passing mechanism. This distributed representation fashion is one of the reasons of flexibility of the control methodology. However, this does not exclude the fact that a behavior implementing a representation might be added to the system and others behaviors learning and operating on the same as well. Additionally, some behaviors might not be internally specified by the program as such, but instead, they might emerge out of the dynamic interaction among all already-existing behaviors and between the robot and its environment or other robots.

This property is essential in such systems as it can expand itself endlessly according to the environment and its interactions. Consequently, these architectures establishes themselves, generally, on models of massively parallel information processing, as it is the case of artificial neuronal networks. These computational models are suitable for applications where there is no *a priori* global knowledge of the world, but rather a set of first level inputs is present. For instance, in order to explain complex behaviors (such as those commonly observed in

animals in their environment) in simpler terms, Gaussier and Zrehen proposed the PerAc (perception-Action) architecture, which uses artificial neural networks a detail description of which is given below [Gaussier 1995].

The complexity, robustness and power of behavior-based systems lays on the way how the behaviors are designed, structured, coordinated and used. Therefore, several and different architectures have been proposed over the last 20 years. Here below, we give a description of three behavior-based architectures that we consider are the most important for the comprehension of such systems and thereby our work.

The Subsumption architecture Contrary to the classic artificial intelligence, which is based on the reasoning and on a centralized system, the new artificial intelligence proposes a hierarchical system inspired by natural processes where every module is directly connected with the system inputs and can generate outputs of the same [Brooks 1990]. Hence, Brooks developed the architecture known as the subsumption architecture, which decomposes complex intelligent behaviors into several « simple » and parallel modules where each is responsible of a single behavior assuring the execution of a different action (see figure 3.11).

In order to choose the most convenient action, the modules are organized on hierarchical layers where each layers has a different priority [Brooks 1986], [Brooks 1991]. The layers are networks of augmented finite state machines composed of a number of states and a set of input and output ports generating each of them a specific behavior.

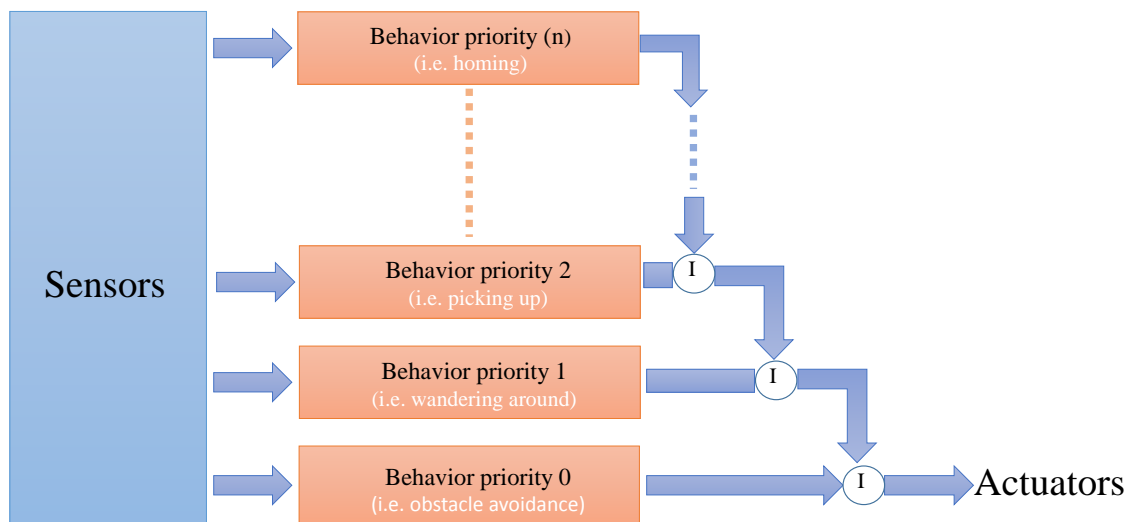


Figure 3.11: Subsumption control architecture

The top layers correspond to abstract tasks, which work to reach the global goal and create viable behaviors by utilizing more concrete and simpler tasks (they subsume lower layers). The lower layers correspond instead to simpler but more “urgent tasks”, therefore their priority precedes that of the top layers. Hence, the lower layers working as mechanisms

of fast adaptation (reflexes) can modify the input of higher layers by means of excitatory and inhibitory knots. A top module is then said to be subordinated by the lower module.

The coordination between each of the behaviors is based on a Priority-based Arbitration technique allowing to decide on the activation of the most active behavior layer when multiple behavior conflicts. For instance, in a case where the lowest layer is «avoid an object» while the second layer is «wander around»; the higher layer «wander around» utilizes the lower-layer competencies to emerge. Hence, by avoiding obstacles the robot is able to wonder around the environment.

Such configuration and functionality, has allowed subsumption robots (Allen, Herbert, Genghis) [Brooks 1990] to react to unpredictable environments by performing behaviors similar to those seen in animals such as insects.

The subsumption architecture showed great success at overcoming problems related to real-time interactions with dynamic environments. However, the memory of the architecture can rapidly be limited by the number of finite states. Certainly, when a reactive action is needed, finite-state machines can be perfect, but when it comes to perform a task requiring some learning and memory, the finite-state machines would fail or would not be sufficient.

Motor schema Architecture The motor schema architecture [Arkin 1987] is another popular example of the first reactive control architectures proposed back in the 80's. It is a biologically inspired approach where motor and perceptual schemas are dynamically connected to one another [Arbib 1981]. The motor schema architecture was proposed as a basic unit of behavior specification that produces an output of each behavior in a vector form. Furthermore, by using a fusion mechanism it merges all the resulting behavior vectors in a manner similar to the artificial potential field concept. Hence, the overall response of the system is achieved by the vector summation of the multiple behaviors as illustrated in figure 3.12. For instance, the generated output allowing the robot to move through a maze, would be a result of the superposition of the behaviors, in this case that of target following and obstacle avoidance.

From the potential field's point of view, the target following task would be represented as an attractive force while the obstacle avoidance task would be considered as a repulsive force where the summation of both forces would coordinate the final action of the robot. However, if attractive and repulsive forces cancelled each other out, the resulting output sum would be null and the robot would remain static. Therefore, in order to overcome this common local minima problem, various solutions have been proposed [Nattharith 2009]. Additionally, the architecture has further been improved in order to achieve more complex tasks [Arkin 1990]. Hence, the autonomous robot architecture (AuRA) added a navigation planner and a plan sequencer, based on finite-state acceptors (FSAs), to the reactive schemas [Arkin 1997].

PerAc Architecture The PerAc (Perception-Action) architecture inspired by the work of [Brooks 1986], [Albus 1991], [Burnod 1990], [Carpenter 1987], [Hecht-Nielsen 1987] and [Edelman 1987] was proposed by [Gaussier 1995] as an organized neural structure that evolves because of the dynamic interaction between the robot and its environment. It has particular properties such as associative memorization, learning by example and parallel processing.

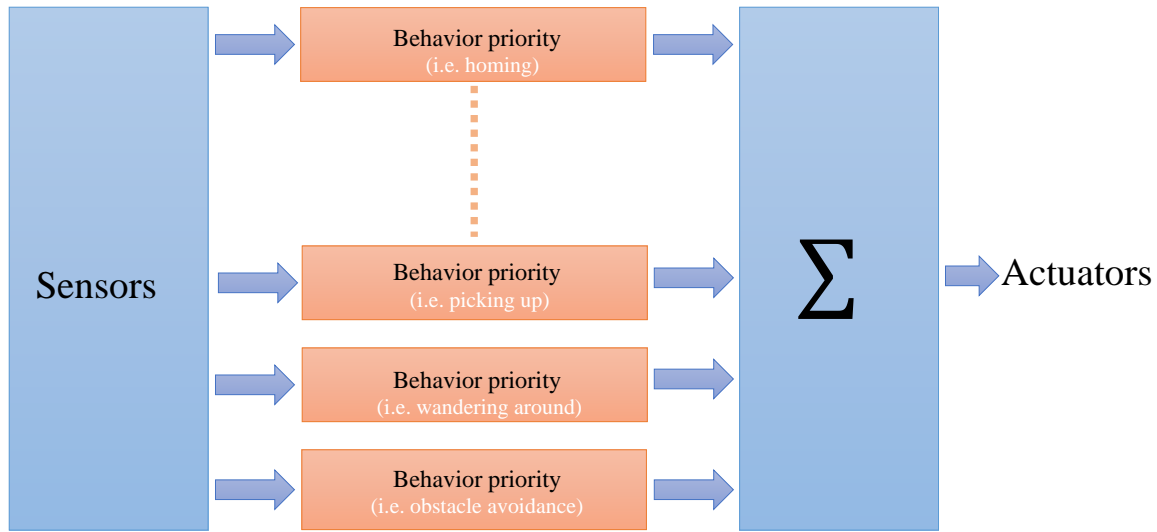


Figure 3.12: Motor schema architecture

The PerAc architecture is composed of two data streams corresponding to perception and action flows (see figure 3.13). The first level uses a reflex mechanism that controls directly the robot's action based on the information extracted from the perceived input. The second level uses a cognitive mechanism performing recognition of the aforementioned perceptive flow and allows learning of the associations between the recognition of a particular shape and the realization of a particular action. Therefore, it is not necessary to have a map or data resulting from a global model of the environment to decide on the actions to undertake.

3.2.2.5 Summary of the approaches

Each of the presented approaches have emerged out of the necessity of enabling robots to autonomously perform a variety of tasks in different domains and applications. While one approach can excel at allowing the robot to perform a given task, the same approach can fail when a different task or goal is required. Therefore, the selection of a control approach depends mainly on the situadness properties of the problem, the type of desired task, the optimality required and the available information. Moreover, it can be tightly linked to the hardware and software robot constraints.

For instance, **deliberative systems** provide an optimal reasoning and planning which is given by an accurate representation of the whole environment. This implies that the environment remains the same; therefore, these systems are ideal for structured and strongly predictable environments, especially in domains where the robot performs repeatedly a given task. Because of the same reason, however, these systems are not suitable for situated robotics.

On the other hand, **reactive systems** give a perfect outcome when it comes to changing environments and where an immediate response and reaction is essential in the performance

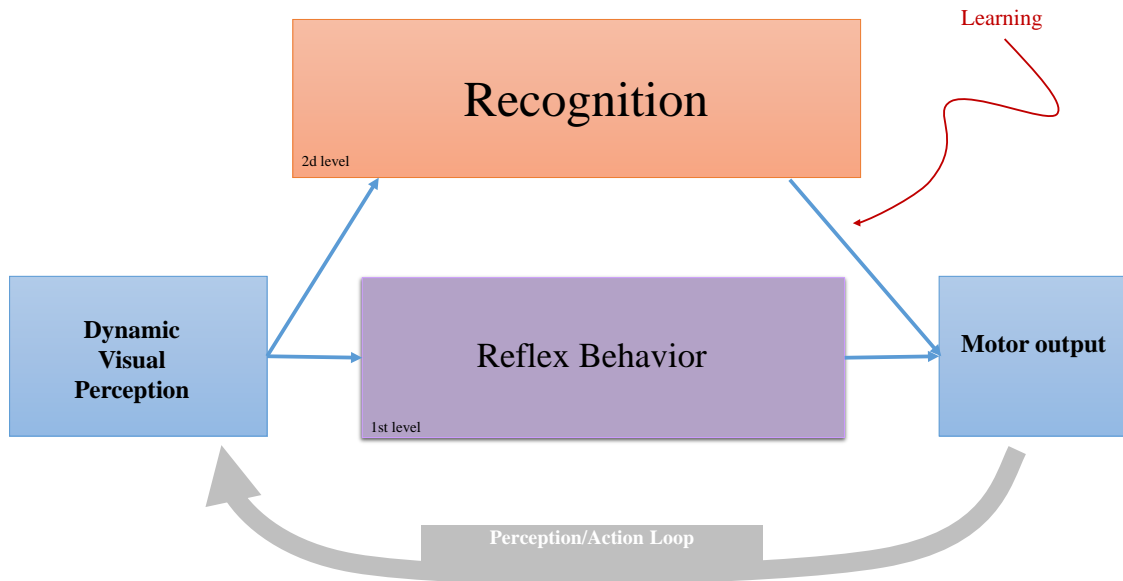


Figure 3.13: PerAc architecture by [Gaussier 1995]

of the task such as obstacle avoidance. Hence, stochastic environments are best dealt with reactive systems. Furthermore, reactive controllers happen to be very powerful in environments and tasks that have been previously characterized. However, their lack of representation of the world and knowledge of the past and future actions, make of these system a problem when planning, learning or memory storage is necessary.

Hybrid systems have shown to be a good solution as they diminish the drawbacks that the other two systems present while getting all the advantages that they both offer. Therefore, they are suited for environments needing internal models and requiring planning in long term and which, real time demands are sufficiently independent of the higher-level reasoning.

Finally, **behavior-based systems** can be said to comprise almost all the advantages that the three other approaches offer plus another related to its behavioral structure: learning and easily adapting to environments that change significantly. Furthermore, their components (behaviors) and their interconnections allow the system to plan, avoid past mistakes and use active representation if necessary. Behaviors are designed at a variety of abstraction levels, facilitating bottom-up construction of behavior based systems. However, the difficulty to implement such an architecture can be huge drawback.

Even though, the ideal control architecture has not yet been developed. Several researches attempt to improve and propose new control architectures by combining, in most of the cases, the best of all these approaches such as optimally reasoning and planning while at the same time quickly responding under dynamic changing environments. Combining behavior-based systems with a global representation of the world can be a good solution. For instance, the

behavior-based architecture (AuRA) facilitates the planning and reasoning by directly using a planner to select behaviors [Arkin 1997]. Similarly, the three-level hybrid architecture 3T uses behaviors in its reactive layer [Bonasso 1995].

Clearing up misconceptions Since describing and implementing behavior-based systems is not always an easy task, they are often misunderstood specially when being compared to other systems. Therefore, it is necessary to clear up some confusions:

Behavior-based vs reactive systems:

Considering reactive systems equivalent to behavior-based systems has been, through the passing of the years, the most common misconception in the robotics literature. Such misunderstanding may not be surprising if one considers only the basic functionality of both systems (as it has been the case most of the time). Truly, as it has been said previously, both systems tightly couple sensing and action into distributed modules. Moreover, they are both presented as being robust in dynamically changing environment when no representation of the world is available. In fact, these common properties are normal as the behavior-based approach has its roots in the reactive approach. However, what strongly differentiate them from each other lays in the fact that contrary to reactive systems, behavior-based systems can store representations and thereby enable reasoning, planning and learning. Reactive architectures, oppositely, lack of an internal state, which makes them incapable of learning and using internal representation.

Behavior-based vs hybrid systems:

Given the use of different modularization strategies by each of these systems, there is often the misconception of considering that one has better expressive capabilities than the other does. In most of the cases, hybrid systems have the upper hand. However, such assumption is erroneous as both of them have the same expressive and computational capabilities of exploiting representations and looking ahead. What makes the difference between both of them is the way of doing that can be suited according to the application domain. For instance, while behavior-based systems dominate the multi-robot control field because of its collection of behaviors, hybrids system dominate that of a single-robot. Only when a task in the single-robot domain is too time demanding, a reactive system is necessary then the behavior-based system would be more appropriate. The set of behaviors within the behavior-based systems allows a robust and adaptive group behavior when working with multiple robots.

Another reason why there might be a misconception is due to the fact that both systems are organized in layers. However, contrary to hybrid approaches, behavior-based do not employ a hierarchical/sequential division and all layers are similar in terms of time scale and representation used. Planning, reasoning and each of the other behaviors use the same mechanisms as the sensing-and-action-oriented behaviors. They provided both low-level control and high-level deliberation whereas in the hybrid approach the layers are drastically opposing to each other (deliberative and reactive).

3.3 Conclusion

This chapter has presented the currently existing control paradigms for building robust and flexible control architectures. Two different viewpoints have been here introduced. Firstly, the functional viewpoint which distinguishes four different paradigms according to the internal functioning and capabilities to act on the environment: reactive, deliberative, hybrid and behavior-based. Secondly, the design viewpoint divided in two different methodologies: top-down and bottom up. They differ in the way of how the sensory data is processed and propagated through the systems as well as how the knowledge is ordered.

We explain their advantages, disadvantages in terms of the needs a robot has for achieving an autonomous navigation.

The RHIZOME architecture we proposed, combines into its neural structure all the above paradigms as it is presented in the next three chapters of part II.

From the **functional viewpoint** the RHIZOME architecture uses an *a priori* knowledge of the environment in order to corroborate the dynamic visual information perceived during navigation. Hence, it is composed of both **deliberative** and **behavior-based modules**. Thus, a **hybrid** architecture. However, the hybrid meaning here, opposes to the currently known hybrid architectures that use an intermediate component to reconcile both representations and to resolve any conflict between their outputs. Conversely, the RHIZOME architecture can be considered as being entirely behavior-based capable of combining two opposing approaches without the need of a coordinator component. Hence, a behavior-based hybrid architecture.

However, it differs from the common behavior-based control architectures in the fact that this architecture does not follow a hierarchical process but instead, each action or behavior is equally important and the resulting action emerges from the interaction with the environment and the internal motivation of the robot.

From the **design viewpoint** the information available from the map is obtained by following a **top-down process** and the action actions of the robot result from a **bottom-up process**.

Part II

The RHIZOME Architecture

RHIZOME 1: Exploring the world with little information

Contents

4.1	General description	91
4.2	Implementation- Rhizome 1 Architecture	94
4.2.1	Overall description	94
4.2.2	Deliberative module – Preconfigured sign sequence	97
4.2.3	Behavioral module–Neural structure	100
4.3	Experiments in real environment	122
4.3.1	Procedure	123
4.3.2	Results	123
4.3.3	Discussion	127
4.4	Conclusion	128

4.1 General description

This chapter introduces Rhizome 1 as the foundation of the Rhizome architecture functioning in a simple deterministic scenario where no unforeseen situations are expected to happen. Such scenario implies that the environment remains unchanged and by consequence, the given *a priori* information (a sequence of navigation signs leading the robot to the final destination) is expected to be found as such in the navigation path during real-time navigation.

Rhizome1 has been built in order to allow the robot to use the navigation signs as reference to navigate towards its final destination, while inferring the directional meaning each navigation sign denotes and learning it for future reference.

The sign sequence is computed beforehand according to the order of appearance of the signs within the path from the starting point to the final destination and it is provided to the robot by means of a command program. All navigation signs used in this work are known by the robot, which means that the robot has already in its database the information describing each sign (see section 4.2.2 for more details). Thus, based on this information, the robot is able to detect one or several signs at a time, as long as they are within the frame of the robot's field of view. Each sign denotes a «directional meaning» (turn right or turn left).

However, in this scenario, the information concerning such directional meaning is not provided to the robot (see Rhizome 2 described in chapter 5, which provides and uses this

information). Therefore, the robot is enforced to deduce the directional meaning by considering the movement leading it to find the next sign of the sequence in the environment. Thereafter, when the same sign appears again, it just follows the implied instruction of the learned sign.

Hence, the robot deduces the directional meaning of a given sign by looking around itself for the position of the next expected sign with respect to its own position when facing the current sign as illustrated in figure 4.1.

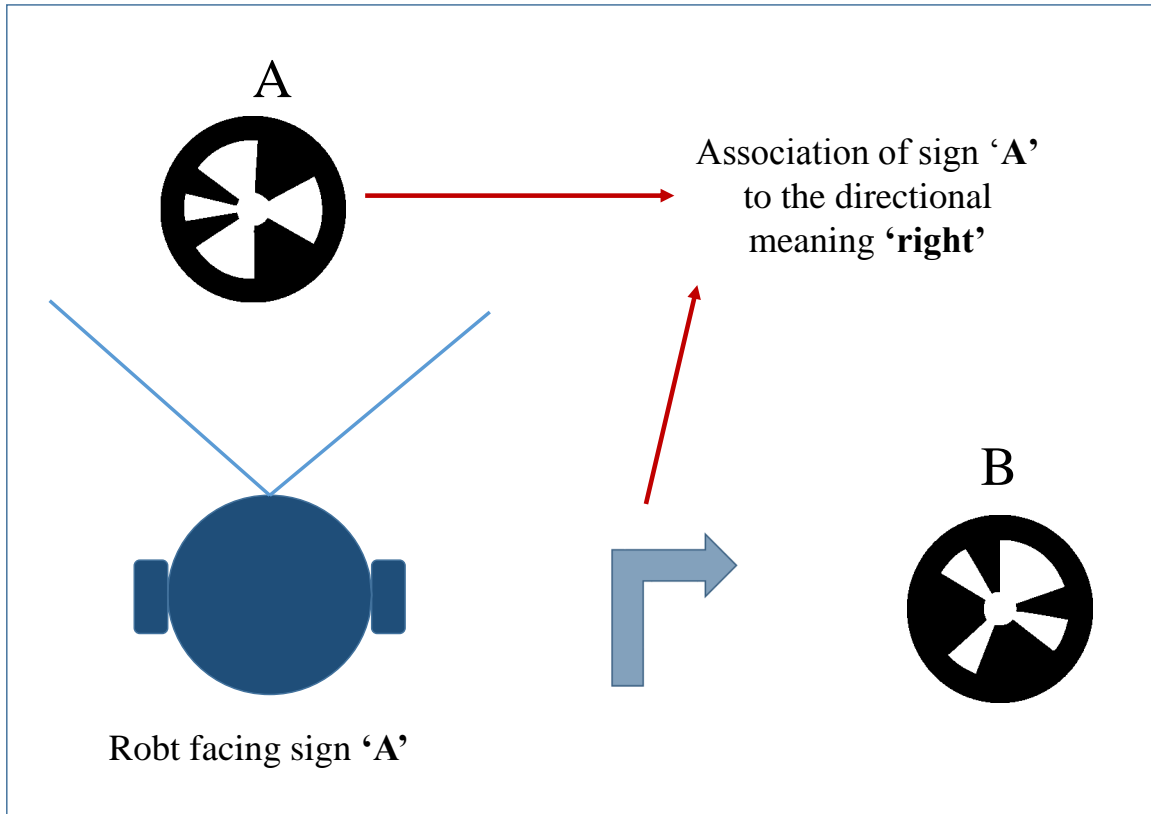


Figure 4.1: Deduction of the directional meaning of the sign «A». The robot looks for the location of the next expected sign «B» with respect to its own position when facing the current sign

Before entering into the details of the implementation and the components of Rhizome 1, let us first have an insight of the overall behavior of the navigation process performed by the robot in this first deterministic scenario (figure 4.2).

The description below which is represented by figure 4.2 summarizes the behavior of the navigation process.

- As explained previously, two sources of information are used as input in the architecture:
 - a) The visual perception information input, which constantly feeds the system in real-time while the robot navigates the environment. When one or several navigation signs appear in the robot's field of view, the signs are considered detected.

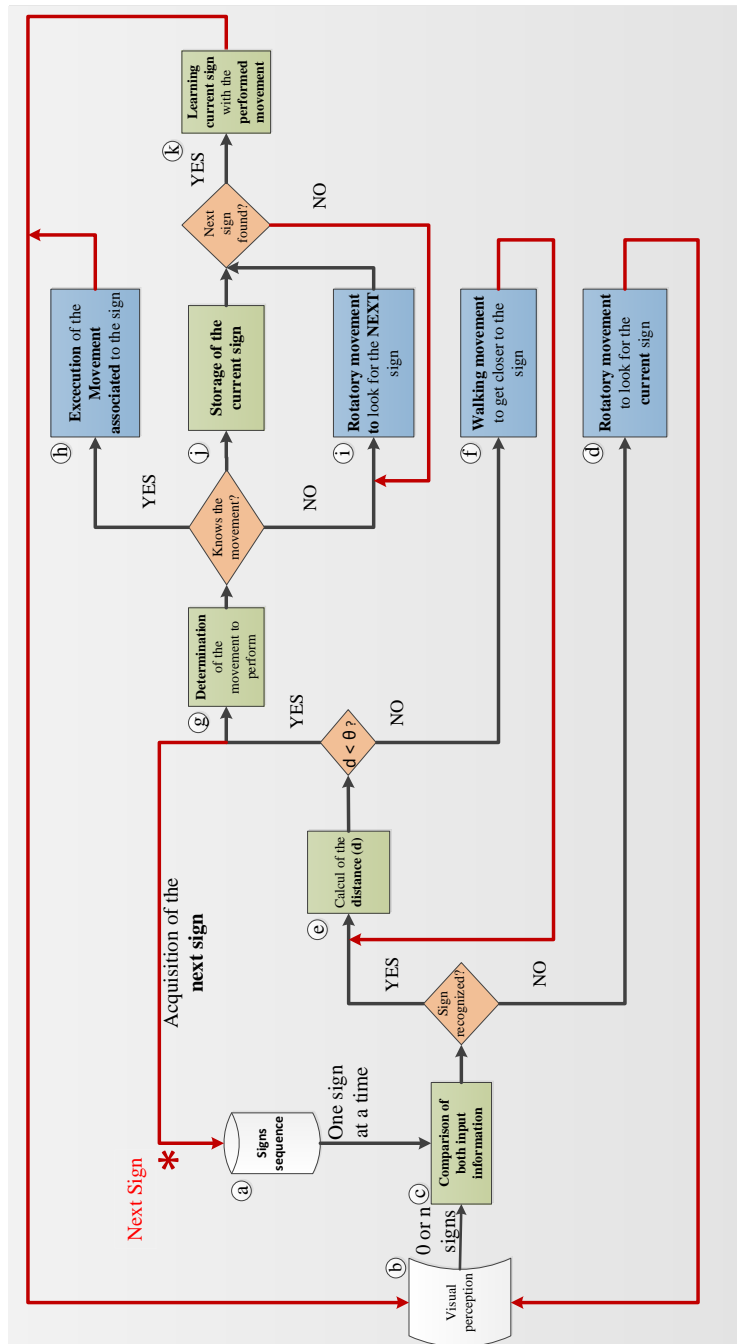


Figure 4.2: Functional diagram of the navigation process behavior allowed by Rhizome 1.

- b) The sign sequence information input, which provides one sign at a time according to what it is expected to be found in the environment. When one of the detected signs matches the current expected sign, the sign is considered recognized. Then,

the sign sequence is scanned to obtain the next sign and provide it as input. The same process is repeated until all signs of the sequence have been recognized in the environment.

- c) In order to arrive to its final destination, the robot looks for each of the expected signs from the sequence within the navigation environment. This is possible by comparing each expected sign to what it sees in real-time.
- d) If the sign is not recognized, it performs rotatory movements around itself in order to look for it until finding it.
- e) Once the sign is recognized (detected sign corresponding to the expected sign), the distance (dis) between the robot and the recognized sign is computed.
- f) If the robot is close enough ($\text{dis} < \text{threshold}$), it can turn around to look for the next expected sign. Otherwise, if the robot is not close yet to the sign, it walks towards the sign in order to avoid premature turns with respect to the intended point of turn for that sign.
- g) Two different actions are possible in order to look for the next expected sign after determining if the directional movement associated to the current sign is known or unknown.
- h) The robot knows the directional movement associated to the current sign: then it simply performs the corresponding movement leading it to the next expected sign. This case is susceptible to happen after the robot has previously seen and learned the said association.
- i) The robot does not know the directional movement associated to the current sign: then, it looks for the next expected sign by performing some rotatory movements.
- j) In the meantime, the robot stores the current sign for a short while.
- k) When the next expected sign is found, it deduces the directional movement out of the performed movement. Then, it associates it to the stored sign and learns the resulting association.

The same process is repeated for each sign until the robot arrives to its final destination.

4.2 Implementation- Rhizome 1 Architecture

4.2.1 Overall description

The overall architecture integrates the signs sequence into an organized neural structure. It is composed of two modules as illustrated in figure 4.3. A **deliberative module**, corresponding to the sign sequence information and a **behavioral module**, which integrates the said sequence information and constantly uses it in order to control online navigation and allow learning of sensory-motor associations.

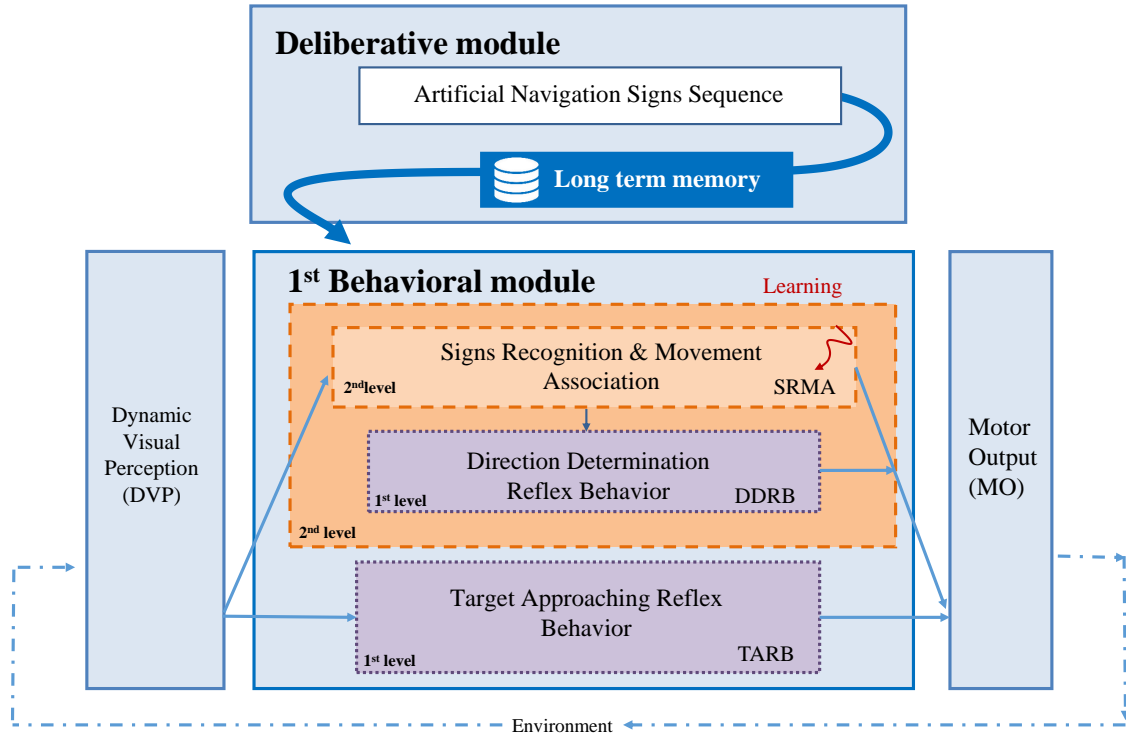


Figure 4.3: Overall view of the Rhizome 1 architecture composed of two modules: Deliberative and Behavioral

- On one hand, the **deliberative** module represented by the top box of figure 4.3 refers to a basic module that provides some important information about the environment before the navigation task begins. More precisely, it consists of a succinct information coding the complete navigation path.

We refer as a basic module in the sense that such *a priori* information is provided directly to the architecture without the use of any complex computing process. In other words, the robot is not entirely autonomous, as it needs to obtain such information by means of an external source.

In this work, the information is primarily represented by a sequence of navigation signs that are expected to be seen by the robot in the real world navigation leading it to the final destination. The sign sequence is computed beforehand according to the order of appearance of the signs within the navigation path from the starting point to the final destination and it is provided to the robot by means of a command program or a voice system. The complete sequence is then stored in the *long term memory* unit of the module to be integrated into the behavioral module.

- On the other hand, the **behavioral** module represented by the lower box of figure 4.3 is based on the PerAc (Perception-Action) architecture as previously explained in chapter 1 and illustrated in figure 4.4.

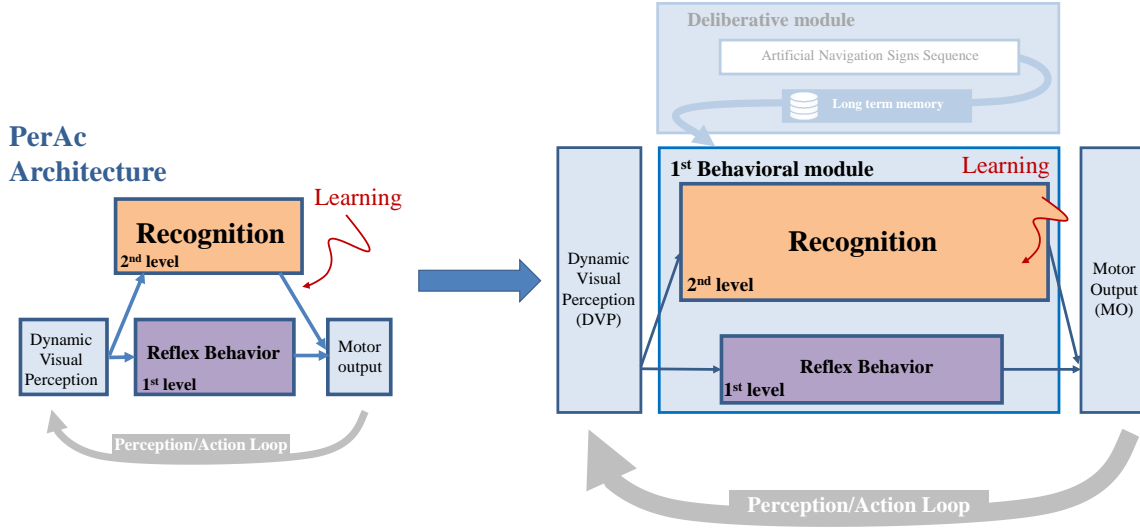


Figure 4.4: PerAc architecture (left) used in the behavioral module of Rhizome 1 (right)

However, the behavioral module of Rhizome 1 differs from the basic PerAc in having a nested PerAc module within its own second level. Hence, it is composed of three layers as illustrated in the behavioral module of figure 4.5 to the left. For the sake of presenting a good visibility of the figures hereafter, the gray arrow indicating the perception/action loop has been replaced by the some pointed blue arrows indicating the interaction with the environment following the same perception/action loop.

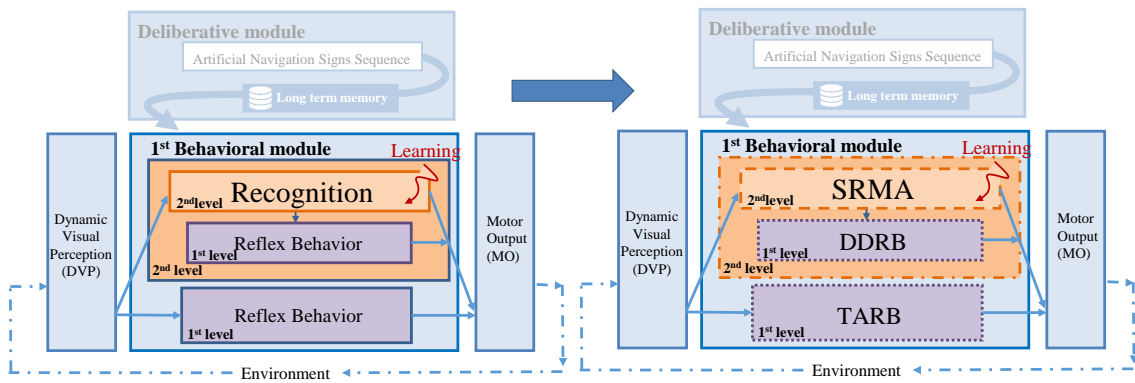


Figure 4.5: Behavioral module composed of a nested PerAc module within its second level (left). As a result, the behavioral module is composed of three layers(Right)

Consequently, by reading from the top to the bottom of the behavioral module of figure 4.5 to the right, each layer is explained as follows.

The first layer of the system (SRMA) is in charge of merging the information coming from the long-term memory and the real-time visual perception of the environment in order to allow the recognition of the expected signs in the navigation environment. Moreover, it is in charge of learning the association between the recognized signs and the movements the robot has performed accordingly.

The other two layers use a reflex mechanism that controls directly the robot's action based on the perceived information of the environment. The second layer (DDRB) determines the direction to be taken by the robot (left or right) by looking for the location of the next expected sign from the pre-captured sequence, while the third layer (TARB) directs an approach towards the sign by keeping it in the center of the robot's vision when the robot is far from it.

The whole system works in parallel and a 'competitive mechanism' allows to decide on the best behavior (among the layers) for controlling the robot according to the stimulus received. This is possible because the neural interconnection is done by either excitatory or inhibitory connections allowing or preventing the activation of neurons respectively. Furthermore, when learning is required, a modulation connection conditioned by a reinforcement signal is used.

The overall architecture follows a perception-action functioning cycle, which means that for every input information coming from the dynamic visual perception of the environment, there is always an action executed which itself alters the perception of the environment for a new process cycle and so on.

Hence, if we take the functional diagram presented in the introduction, each module intervenes on each of the following actions as illustrated in figure 4.6.

4.2.2 Deliberative module – Preconfigured sign sequence

This module is in charge of storing the sequence of navigation signs that are expected to be seen by the robot when navigating the environment.

The navigation signs used in this work consist of artificial landmarks designed with a predetermined contrast, size, and shape, so they can easily be recognized with respect to more complex objects in the environment.

Since the whole architecture has been implemented and tested in the Aldebaran Robotics' NAO humanoid robot, we decided to use the landmark detection system already implemented by the company. Indeed, the «Naoqi» framework that comes with the Nao platform allows the robot to recognize special landmarks called «Naomarks» which are characterized by white triangle fans inside black circles. As illustrated in figure 4.7), each Naomark differs from one another in the size and location of the inner white fans. Moreover, each of them has a unique tag number «Mark ID » serving as its identifier.

The recognition system has been built such that it is possible to detect distinctively each of the Naomarks (figure 4.8). The detection system is able to obtain some important information in terms of the camera angles of the robot as follows:

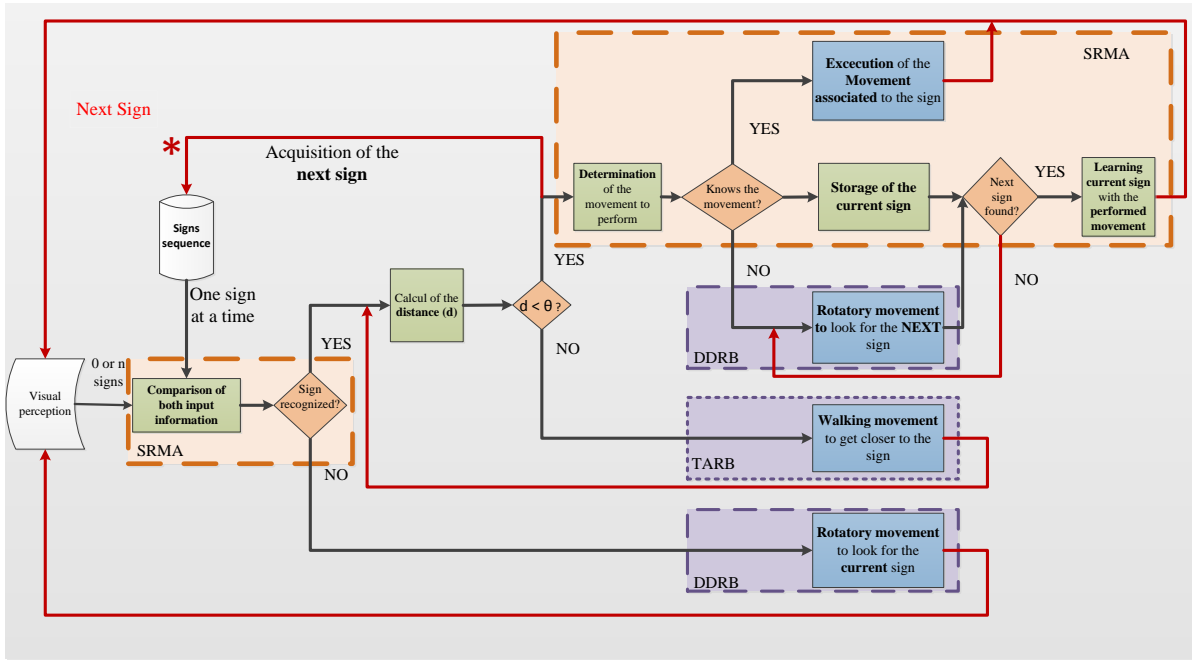


Figure 4.6: Functional diagram of the navigation process behavior allowed by Rhizome 1 with the architecture layers acting on each functional decision.

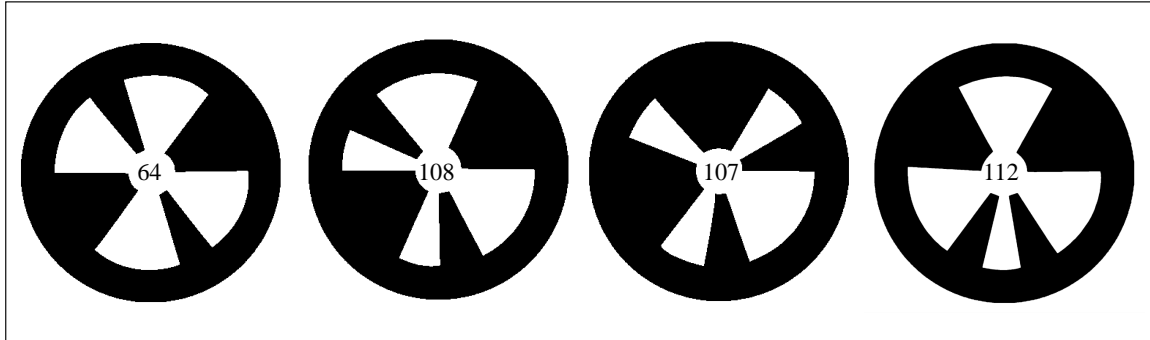


Figure 4.7: Example of some Naomarks used in this work as the navigation signs

- In the case of the detection of N naomarks, the variable structure consists of two fields. $[[\text{TimeStampField}][\text{Mark_info}_0, \text{Mark_info}_1, \dots, \text{Mark_info}_{N-1}]]$ with:
 - $\text{TimeStampField} = [\text{TimeStamp_seconds}, \text{TimeStamp_microseconds}]$. This field is the time stamp of the image that was used to perform the detection.

- Mark_info = [ShapeInfo, ExtraInfo]. For each detected mark, we have one Mark_info field.
 - * ShapeInfo = [0, alpha, beta, sizeX, sizeY, heading]. alpha and beta represent the Naomark's location in terms of camera angles - sizeX and sizeY are the mark's size in camera angles - the heading angle describes how the Naomark is oriented about the vertical axis with regards to NAO's head.
 - * ExtraInfo = [MarkID]. Mark ID is the number written on the naomark and which corresponds to its pattern.
- When no naomarks are detected, the variable is empty. More precisely, it is an array with zero element, (ie, printed as «[]» in python).

Therefore, it is possible to keep in memory the information of each Naomark in order to compare it to what it is currently being perceived by the robot while navigating within the environment. Additionally, by setting some threshold values according to the desired task, it is possible to set the distance at which we consider the robot is close enough to the sign.

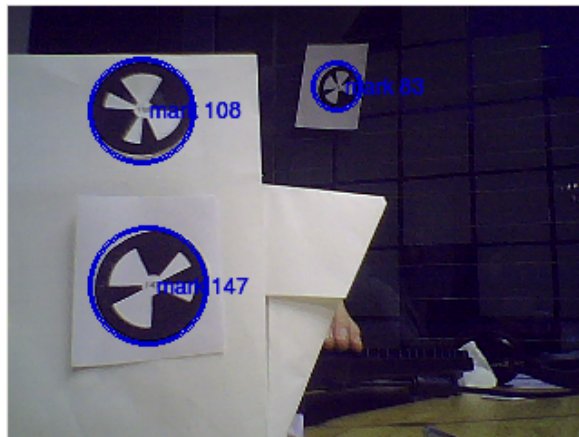


Figure 4.8: Example of some Naomarks detected in the environment by the Nao Robot [figure extracted from Aldebaran's documentation]

4.2.2.1 Performance and Limitations

The following data are directly extracted from Aldebaran's documentation.

Lighting: the landmark detection has been tested under office lighting conditions (i.e., under 100 to 500 lux). As the detection itself relies on contrast differences, it should actually behave well as long as the marks in the input images are reasonably well contrasted.

Size range for the detected Marks:

- Minimum: ~ 0.035 rad = 2 deg. It corresponds to ~ 14 pixels in a QVGA image
- Maximum: ~ 0.40 rad = 23 deg. It corresponds to ~ 160 pixels in a QVGA image
- Tilt: ± 60 deg (0 deg corresponds to the mark facing the camera)
- Rotation in image plane: invariant.

In order to illustrate in a simple form the use of these signs through this work, each sign will be referenced from now and on, by a letter instead of a number as the identifier «IDMarker» as shown in figure 4.9.

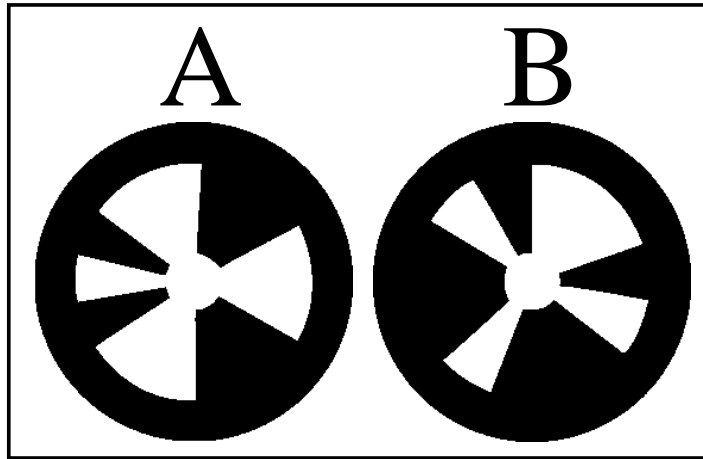


Figure 4.9: Exemple of Naomarks referenced by tag letters «A» and «B»

Even though, the robot is able to detect the Naomarks at any moment, it is necessary to give a recognition directive in the navigation context. Therefore, the actual recognition task is handled by the behavioral module described just below (section 4.2.3). In fact, only when the expected landmark from the sequence extracted from the map is detected in the environment, it can be considered to be recognized (refer to Sign Recognition and Movement Association (SRMA) layer). Additionally, the location of the landmark with respect to the camera frame can also be computed (refer to Target Approaching Reflex Behavior (TARB) layer).

4.2.3 Behavioral module–Neural structure

The behavioral module is composed of several neural groups, which each one itself, is composed of a certain number of neurons. These neural groups are interconnected according to two types of links: a *one-to-one link*, where each neuron in a given group is connected to only one and unique neuron from another group; and a *one-to-all link*, where each neuron in a given group is connected to all neurons from another group (see figure 4.10).

When the *one-to-one link* is used, the information received by a given neuron of the second group is transmitted by a single neuron of the first group independently from the information of the others neurons of the first same group. On the contrary, when the *one-to-all link* is used, the information received by a given neuron is obtained from the addition of the information of all neurons of the first group and depending on the task; a learning process can take place.

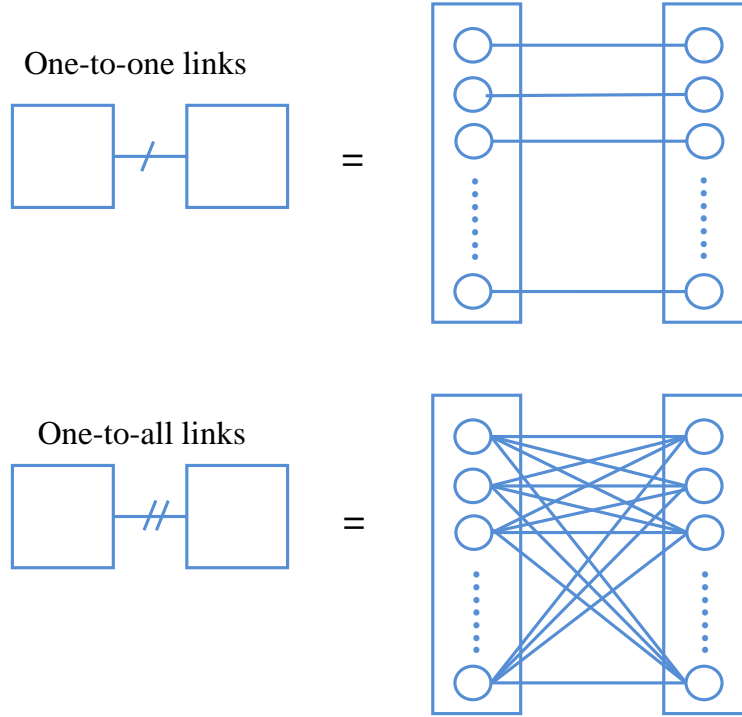


Figure 4.10: Types of links used to interconnect the neurons. The *one-to-one link* connecting each neuron in a given group to only one and unique neuron from another group; and the *one-to-all link* connecting each neuron in a given group to all neurons from another group.

Each neural group is in charge of encoding a given task and by connecting some of them in a sequential order; it has been possible to construct each of the three horizontal layers previously mentioned.

As illustrated in figure 4.11, **both *a priori* information stored in the Long Term Memory of the deliberative module and dynamic visual information (DVP)** are the input of the behavioral module. While the *a priori* information only feeds the SRMA recognition layer (second level), the dynamic visual information feeds simultaneously both layers: The TARB reflex layer (first level) and the SRMA recognition layer (second level), which transfers itself the information to the DDRB reflex layer (first level).

Each of the three layers process the information simultaneously and independently from the other layers, and they all converge towards the same **Motor Output(MO)** with a

resulting movement to be performed. However, only one movement is allowed to be executed at once. Such decision is made by considering the input information, the internal process of the neural structure and the activation of the «proximity sensor» neural group which directly links the dynamic visual information to the **Motor Output(MO)**.

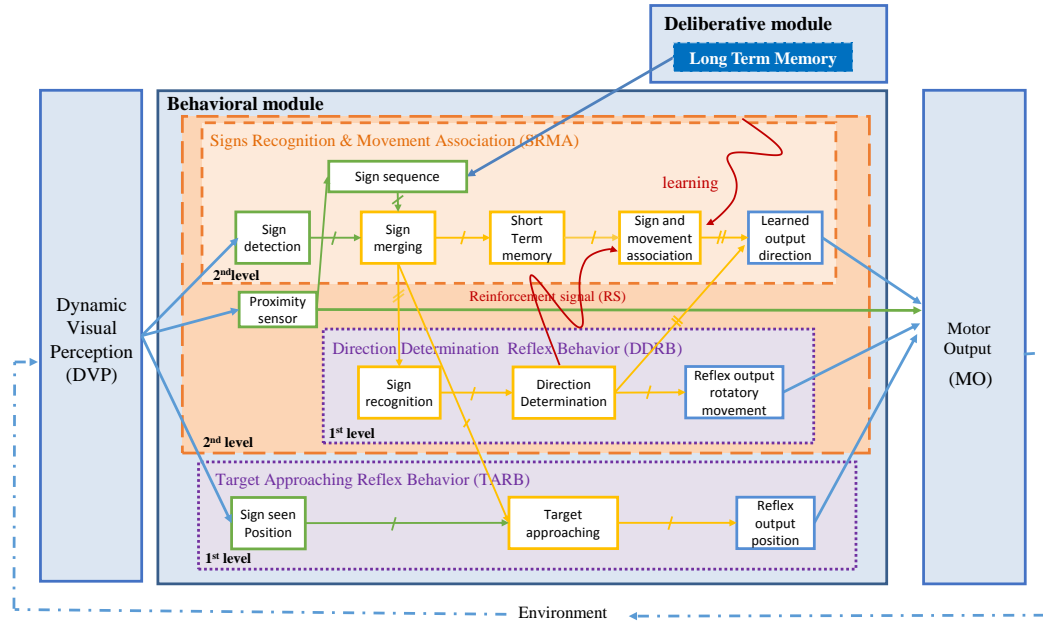


Figure 4.11: General view of the three layers in the behavioral module

In order to understand the functionality of the overall behavioral module and its three layers, this section has been divided in four parts. First a general description of each of the three layers is provided, followed by a more detailed description explaining each one of the neural groups composing them. Thereafter, the layers convergence in the motor output group is explained, to finish up with a detailed description of the layers interaction according to a given example.

4.2.3.1 General description of layers

After the complete sequence of signs is given to the robot, it stores it in its long-term memory to use it. Then, at the start of the exploration, the robot may or may not know the meaning of each sign in terms of the instruction it represents with respect to way finding. The architecture is designed such that if the directional meaning of the sign is unknown, a reflex exploratory behavior gradually leads it to the correct direction and then the association between the sign and the movement performed is learnt (see DDRB). The learning is conditioned by a reinforcement signal which information is transmitted by a modulation connection to the SRMA layer. Hence, if the same sign appears again and it has already been associated to a particular movement, the robot knows which direction to take and it executes the related movement i.e. turn left or right (see SRMA). Additionally, the architecture also performs a

target approaching behavior when the robot is far away from a sign in order to be able to read it (see TARB).

Signs Recognition and Movement Association (SRMA): Once the robot begins exploration, this level enables the robot to perform a movement based on the combination of the *a priori* information stored in the sign sequence group and the dynamic visual information perceived from the robot's camera stored in the *sign detection group*. When exploring the environment two scenarios are possible: the expected sign in the sequence obtained from the deliberative module is recognized or not. In the former case, if the sign has already been associated with a particular movement, the robot executes directly the related movement i.e. turn left or right defined by the *learned output direction group* (see (1) in figure 4.12).

On the other hand, if the sign has been recognized but not associated with a particular movement yet or it has not been recognized at all, a reflex rotatory movement is triggered in the DDRB layer (see (2) in figure 4.12) in order to look either for the next expected sign in the sequence or for the current sign respectively. If the recognized sign has not been associated with a movement yet, the *short term memory group* stores the value of the current sign while the next expected sign is being looked for.

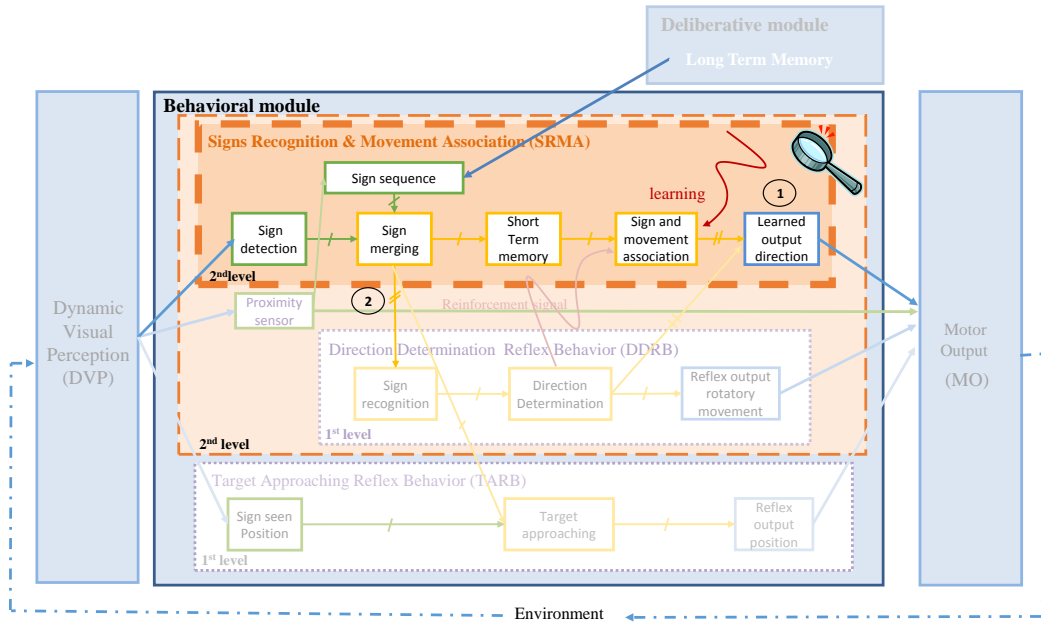


Figure 4.12: General view of the Sign Recognition and Movement Association (SRMA) layer

Direction Determination Reflex Behavior (DDBR) This layer is in charge of making the robot explore the environment by rotating in one place (to its left by design) using small reflex movements in order to look for the expected sign in the environment. This occurs in one of the following cases:

1. The received visual input from the camera does not correspond to the expected sign. In this case, the robot continues to search for it using the aforementioned rotational reflex movements. If the sign is found, the SRMA level (explained above) and TARB level (explained below) are activated. This case is likely to happen only at the very beginning of the exploration to locate the first sign of the path.
2. The expected sign is recognized but it has not been associated yet with a specific movement. In this case, the robot searches for the next expected sign from the pre-captured sequence by performing rotational movements. Once this next sign is found, the angle of rotation undergone is allocated to the current sign as its associated movement in that direction (left by default). If this angle is greater than 180° , the movement to be associated is a turn in the opposite direction (right) (see (3) in figure 4.13). Thereafter, the reinforcement signal is activated so as to learn the association in the SRMA level (see (4) in figure 4.13).

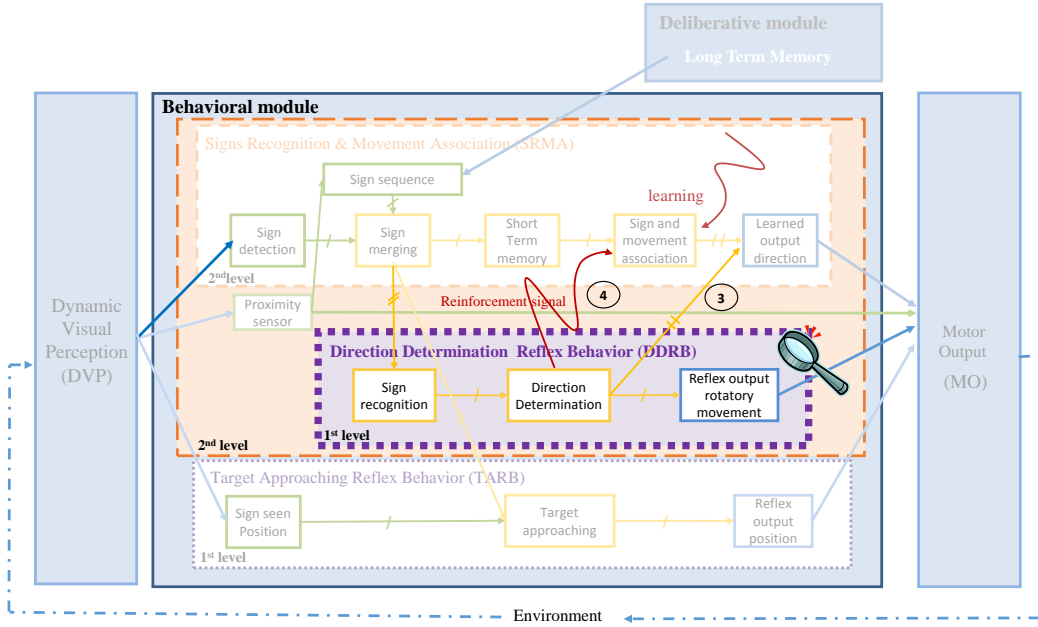


Figure 4.13: General view of the Direction Determination Reflex Behavior (DDRB) layer

Target Approaching Reflex Behavior (TARB) When the robot is far from the sign, this level allows the robot to direct an approach towards the sign by keeping it in the center of the robot's field of vision. If, for instance, the sign is situated at the left side in the robot's visual space, the movement to be performed, is some steps ahead towards the left as illustrated in figure 4.14. It is important for the robot to approach the target signs to avoid premature turns with respect to the intended point of turn for that sign.

Since the input of this layer is the visual perception of the environment before being compared with the sign sequence information, it computes the position of any detected sign

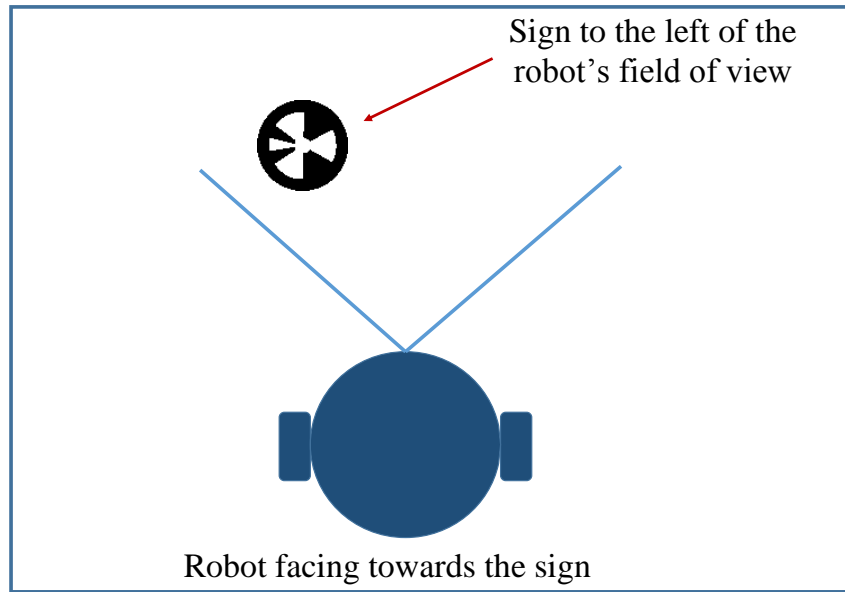


Figure 4.14: Exemple of the direction towards which, the robot needs to turn to keep the sign centered while approaching the sign. In this case towards the left since the signs is to the left side of the robot's field of view

regardless if it has been recognized or not (see (5) in figure 4.15). However, the movement can only be executed if the sign has been recognized, which event is triggered by the SRMA layer and connected directly to the *target approaching group* (see (6) in figure 4.15).

4.2.3.2 Detailed description of layers

Each of the three layers presented above, is composed of a number of groups of neurons in charge of performing specific tasks according to the input information and activation threshold. Thus, the section below, presents each of the layers by first giving a general overview of the neural groups composing them and then detailing individually the composing neural groups.

For a better understanding, the neural groups are presented according to three types of neural units: The **input units** in charge of receiving the input information, the **internal units** allowing processing the input information and the **output units** permitting the robot to execute the motor action according to the processed information.

Be aware that this unit distinction should not be seen similar to the unit distinction exposed by some of the most commonly known neural models such as RNN or CNN, which use hidden units as explained in the introduction section 1.3.1.

Signs Recognition and Movement Association (SRMA): This level is composed of eight neural groups as shown in figure 4.16 and explained below. Each group (excepting from

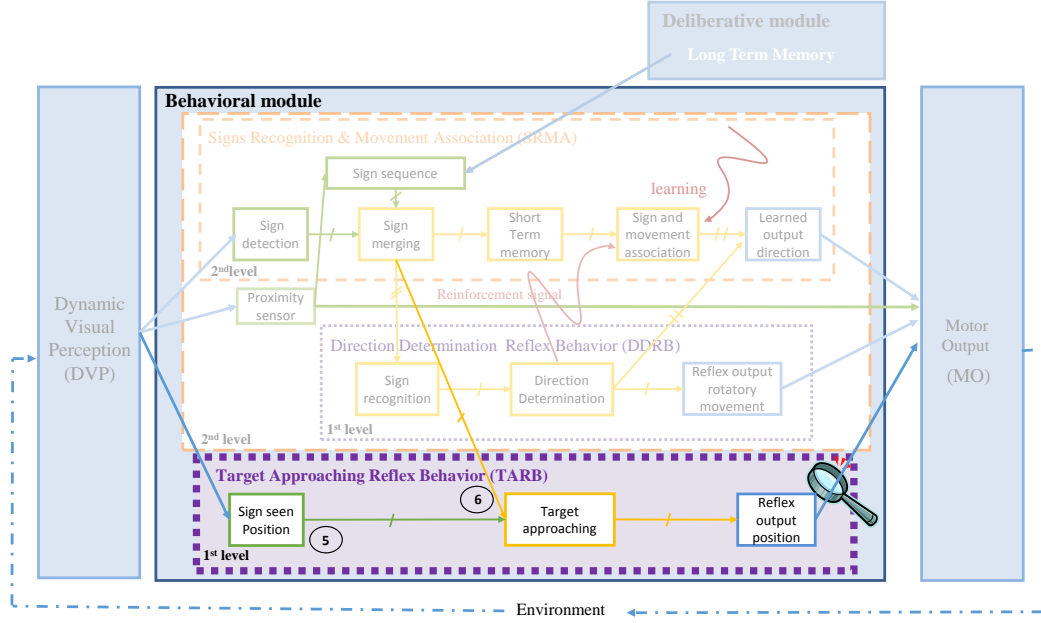


Figure 4.15: General view of the Target Approaching Reflex Behavior (TARB) layer

the *output direction group* and the *reset group*) has as many neurons as the total number of detectable signs known by the robot, each neuron representing a unique sign. Whereas the *reset group* is composed of a single neuron, the *output direction group* is composed of two neurons for left and right movements respectively.

Input units

Both, the *Sign detection group* and the *Sign sequence group* receive the input information directly from the visual perception and the deliberative module respectively. In order to calculate the potential and the activation function of these two groups, two array database have been created and connected to the neural groups respectively. They both store the same information concerning the identification tags of all signs known by the robot that it can detect in the environment. The order of their storage in the arrays correspond to the order of the corresponding neuron in the neural groups (see figures 4.17 and 4.18).

Sign detection group: As the robot interacts with its environment, dynamic visual information is constantly fed into the neural group. However, it is only activated if one or more signs appear in the robot's view activating to the maximum value '1' their corresponding neuron.

The potential value of all neurons is zero by default and its value is calculated as follows:

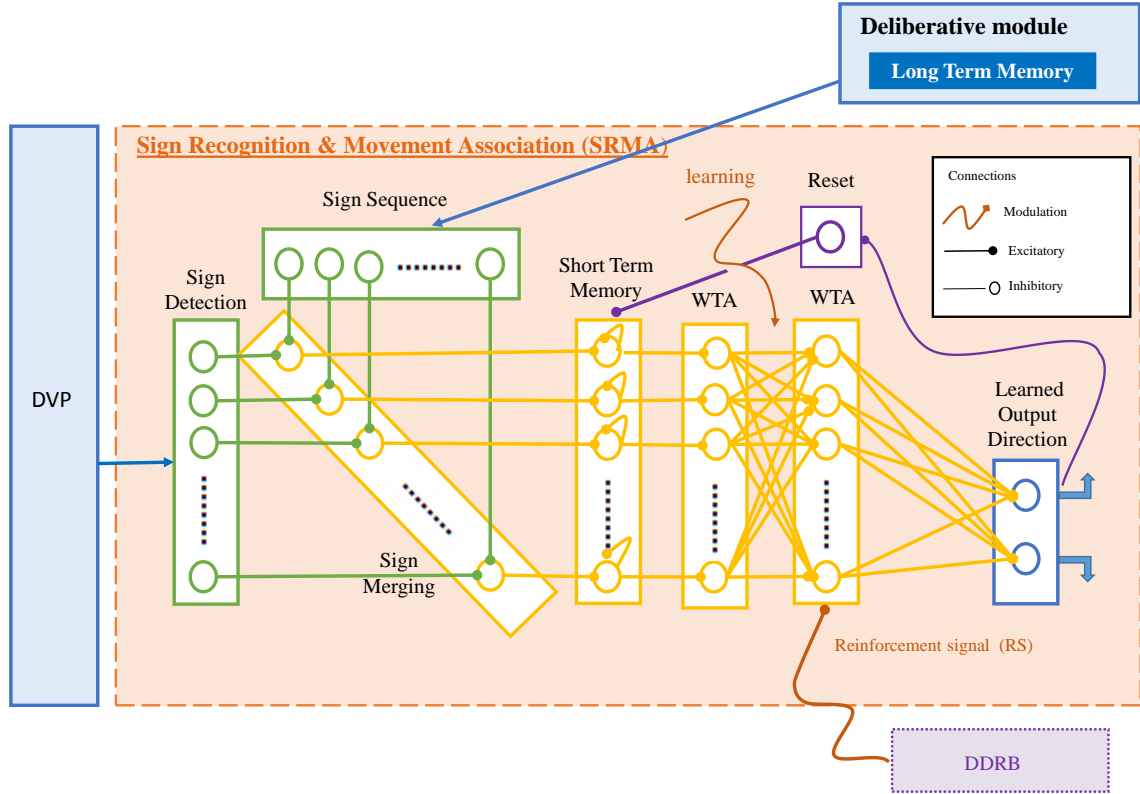


Figure 4.16: Detailed view Signs Recognition and Movement Association layer (SRMA)

Input: one or several signs in the robot's visual perception

Output: potential of the corresponding detected neurons equals to one initialization;

```

for each signi in the ArrayDatabase do
  if signi is equal to the perceived sign in the environment then
    Potential value of the corresponding neuron,  $p_i = 1$ ;
  end
end

```

Algorithm 1: Potential value computation of the neurons of the sign detection group

Then, the activation a_i of the neurons is defined by a linear function.

Sequence sign group: This neural group is fed by the sequence of signs provided by the deliberative module (Long Term Memory unit). Only one sign from the sequence is transferred as the expected one at a time and its corresponding neuron becomes activated at its maximum value '1'. Once the robot is close to the sign, the sequence is scanned so as to obtain and transfer the next expected sign (see figure 4.18). This situation is repeated until the end of the sequence.

The potential value of all neurons is zero by default and their value is calculated as follows:

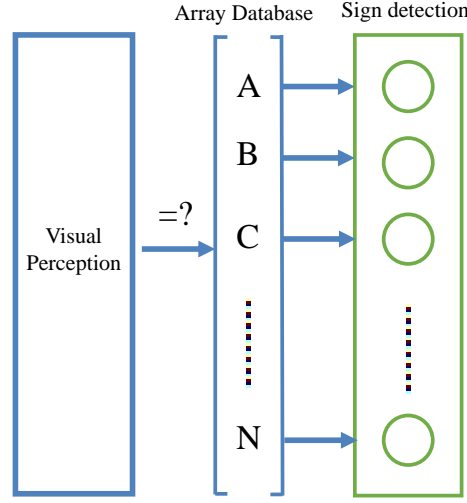


Figure 4.17: Activity of the sign detection group. The dynamic visual information is constantly fed into the neural group then when one or more signs appear in the robot's field of view its corresponding neuron gets activated to the maximum value.

Input: one expected sign at a time from the sign sequence

Output: potential of the corresponding neuron equals to one initialization;

sign_i = first sign_i from the Array Database;

sign = first sign from the Long-term Memory Sign Sequence;

while there is a sign in the Long-term Memory Sign Sequence **do**

if sign is equal sign_i **then**

 | Potential value of the corresponding neuron, $p_i = 1$

end

if the robot is close to the sign **then**

 | Obtain the next sign from the Long-term Memory Sign Sequence;

end

 obtain next sign_i from the Array Database

end

Algorithm 2: Potential value computation of the neurons of the sign sequence group

Then, the activation a_i of the neurons is defined by a linear function.

Internal units

Once both input information have been encoded in the two corresponding neural groups, it is necessary to corroborate if among all the information coming from the visual perception of the environment, there is one sign that would correspond to the same expected sign from the sequence. Consequently, the information should be merged and compared.

This task is performed by the *sign merging group* that sends the result to the other

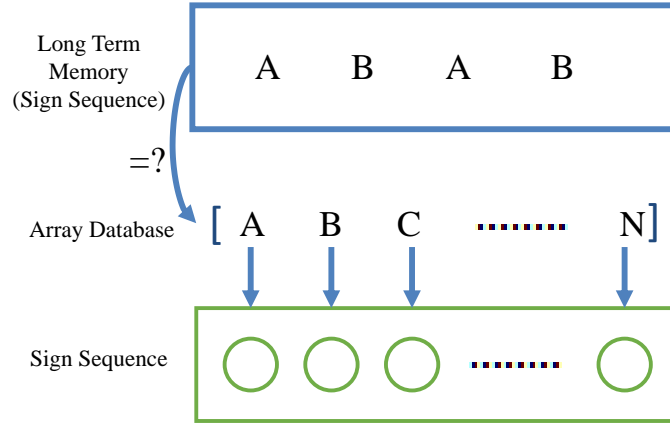


Figure 4.18: Activity of the sequence sign group. The sign sequence provided by the Long Term Memory in the deliberative module is fed into the neural group. Only one neuron is activated at a time corresponding to the expected sign.

neural groups in the architecture. When the comparison results positive (a sign is recognized) the corresponding neurons are activated, allowing the movement, to which it has been associated with in the WTA group by a learning procedure, to be performed in the output groups. However, since the movement might not have been learned yet, the robot needs to look for the next expected sign. This implies that a new comparison of both input information needs to be done, with the only exception that this time, the sign from the sequence happens to be another one. However, if the current sign is not stored somewhere in the memory of the architecture, the new expected sign will override the activity of the current neuron. Therefore, a *short term memory group* placed between the *sign merging and WTA group*, is used to store the activity value of the current neuron sign while the next sign is being looked for.

A explanation of these groups is given as follows.

Sign merging detector group: This neural group fusions and compares both input information coming from the two input neural groups (see figure 4.19). Its activation is defined by a Heaviside function whose threshold value allows the activation of the neuron whose both inputs values are equal to one. Since all the neurons encode a different sign, only the neuron corresponding to the expected sign will be activated.

Short term memory group: It stores the activation value of the detected current sign. The value increases as long as the sign is within the robot's field of view while the robot approaches it. The further the robot is from the sign, the higher the value is.

If v is the activity of the recognized sign neuron i , then its corresponding short-term memorization u_i (time constant τ) as illustrated in 4.20 can be computed as in equation 4.4

$$\tau \cdot \frac{du_i(t)}{dt} = \alpha v_i(t) - \beta r_i(t) \quad (4.1)$$

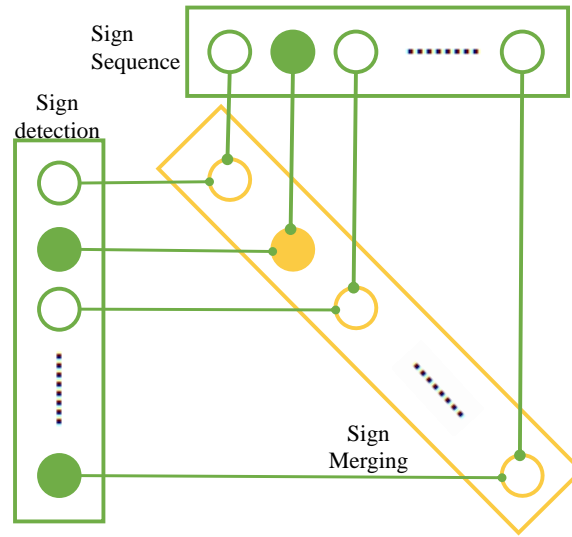


Figure 4.19: Activity of the Sign merged detector group. The neuron whose two input values are equal to one gets activated defined by a Heaviside function

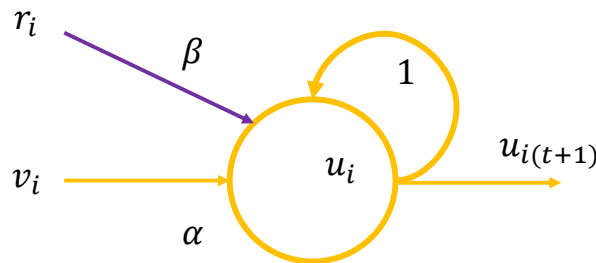


Figure 4.20: Activity of a neuron u_i of the short term memory group defined by equation 4.4

Where, r_i is the activity of a reset neuron allowing setting the values to 0 when the sign-movement association has already been learnt. The variables α and β are their associated weights with α really small and $\alpha \ll \beta$.

The resulting $u_i(t+1)$ corresponds to the potential value and its activation is computed by using a ramp function.

WTA group: a competitive mechanism «winner-take-all» enables the neuron with the highest activation value to stay active whereas all the other neurons are set to zero. The resulting activated neuron represents the current sign to be associated to a particular action.

The interconnectivity is made in such a way that it allows the learning of the said association conditioned by the activation of a reinforcement signal, which is set in the reflex level (see DDBR layer). The synaptic connection of the neurons is then modified based on the following equation 4.2

$$\Delta W_{ij} = \varepsilon \cdot \Delta x_i \cdot \Delta y_j \cdot R \quad (4.2)$$

Where, x_i is the input neuron, y_j is the current neuron, W_{ij} is the weight of connection between x_i and y_j , ε is the learning rate and R the reinforcement signal that is only set to '1' if an association sign-movement needs to be learned, otherwise it is equal to '0'.

However, since the values resulting from the memory group and those of the synaptic weights are quite small, it is likely that more than one neuron or the wrong neuron would be activated. This case is susceptible to happen if for instance, the robot is already close to the sign and it looks around for the next expected sign. Then, the activity value of the neurons corresponding to current and the next sign will be the same, which would cause an ambiguity when learning the association.

Therefore, in order to allow a more robust choice by using integers values (1 and 0) instead of directly using the float values of the memory group, two WTA groups are here used as illustrated in figure 4.21.

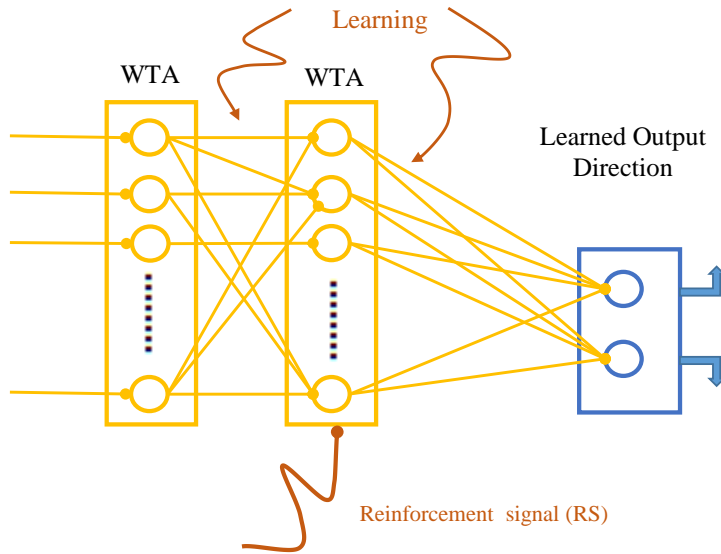


Figure 4.21: Activity of the WTA group. Here, two WTA groups are used in order to avoid causing ambiguity when learning the association.

Reset group: As its name indicates, it resets the values stored in the short-memory group to zero once the association between the sign and the movement has been learned and a new sign needs be processed.

Output unit

Output Direction group: It receives input from both levels (the current one and DDBR seen in the next section). If the sign has already been associated to a particular movement, its corresponding neuron is activated and sent directly to the motor output. Otherwise, if the reinforcement signal is activated in order to learn the association between the movement triggered by DDBR and the current sign.

Direction Determination Reflex Behavior (DDBR) This layer is mainly composed of eight neural groups as illustrated in figure 4.22. It receives the information coming from the SRMA layer. Therefore, there are not neural input units.

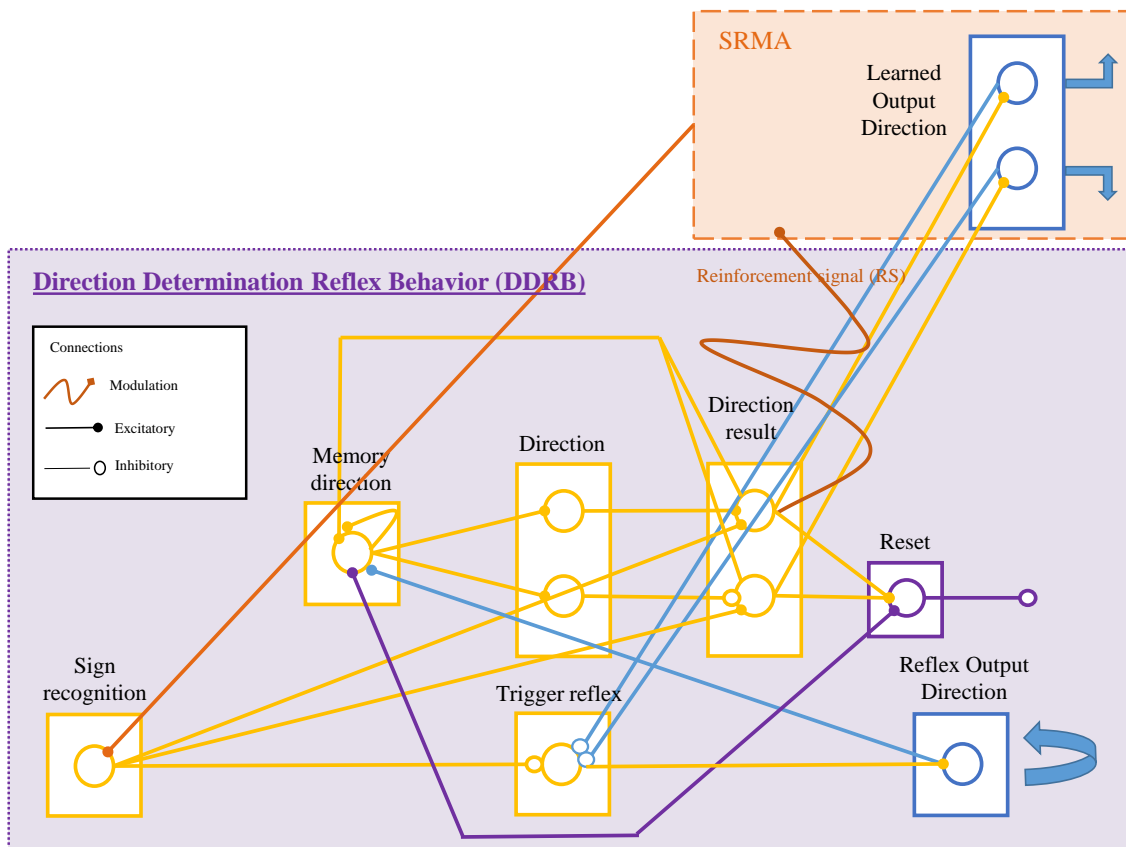


Figure 4.22: Detailed view of the Direction Determination Reflex Behavior layer (DDBR)

Internal units

When the comparison performed by the **sign merging group** in the SRMA layer results negative (sign not recognized), this layer activates its neural groups allowing the robot to perform rotational movements to compute the directional meaning of the current sign by

looking for the next expected sign.

Sign sensor recognition: Composed of a single neuron, this group is connected to all neurons of the *merging group* from the SRMA layer. It serves as an indicator allowing knowing if a sign has been recognized or not. Thus, it only takes one activated neuron from the *merging group* to stimulate the activation of this group defined by a sign function.

Trigger Reflex group: This single-neuron group triggers or inhibits the reflex movements. Since the *sign sensor recognition group* and the *output direction group* are linked to this group through an inhibitory connection, the Heaviside function allows to activate it when no sign has been recognized and when no associated movement is known.

Memory Angle group: In order to calculate the total angle of rotation, each turning-angle is stored and then added to itself as many times as it is required to find the next sign. Once the sign is found, the total angle is transmitted forward and then reset to zero for the next calculation. The total angle of rotation is computed by following the same equation 4.4 of the *short-term memory group* as illustrated in figure 4.20.

Direction group: The total angle of rotation calculated in the previous group is compared to a threshold value so as to compute the activation of the current neurons by using the sign function. However, as each neuron represents either a left or a right movement, the resulting activation output of one neuron excludes that of the other.

Direction Result group: It takes as inputs the results of the *direction group* and the *sign sensor recognition group*. Therefore, only when the next sign has been recognized, the activity value of the neuron from the direction group triggers the activation of its corresponding neuron in the *direction result group*. Then, the result (movement to be associated to the current sign) is sent to the *Learned Output Direction group* of the SRMA layer and the reinforcement signal \mathbf{R} is set to '1' in the WTA group of the same layer so as to allow the learning of the association between the current sign and the resulting movement.

Output units

Reflex Output Direction group: Whenever the *trigger reflex group* activates this single neuron group, it sends the information to the *motor output* so as to perform small leftwards rotational reflex movements.

Learned Output Direction group: This neural group of the SRMA layer receives as input the resulting values of the *Direction result group* of the current DDRB layer and since the reinforcement signal is set to '1', the association between the current sign and the movement (neuron activated in this group by the Heaviside function is learnt. However, since the *reflex output rotatory group* has already performed the movements leading the robot from the current sign to the next, a *reset group* inhibits the movement to be executed in the *Motor Output (MO)* (see figure 4.23).

Target Approaching Reflex Behavior (TARB) This layer receives as input the information coming from the visual perception. Alike the SRMA layer, the TARB layer encodes the information of the total number of detectable signs known by the robot. However, since it computes the position of the detected signs within the robot's field of view, the number

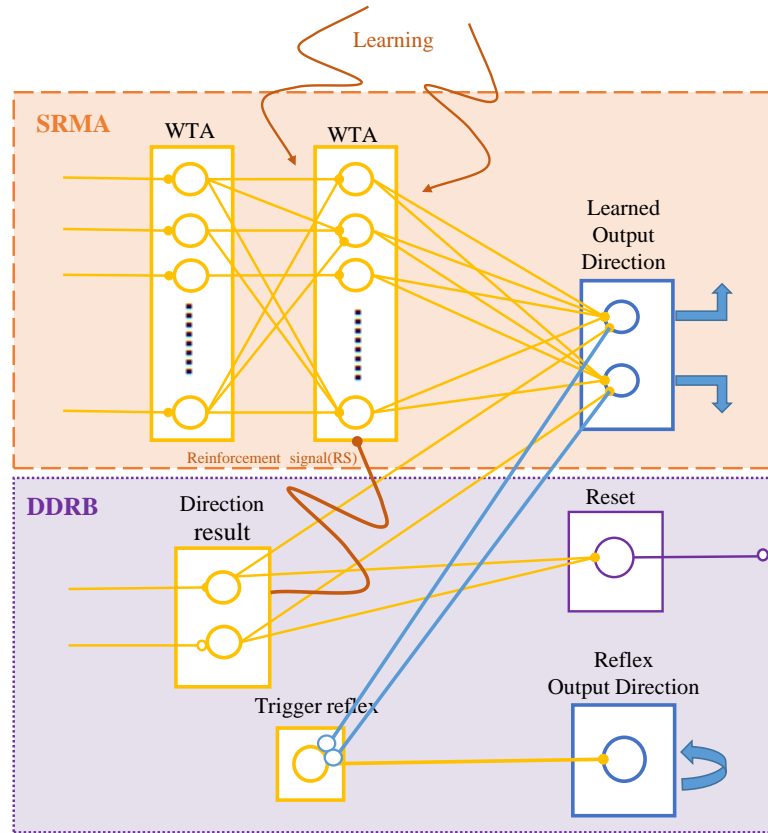


Figure 4.23: Activity of the Learned Output Direction group. It receives the input from both WTA groups of the SRMA layer and Direction result group of the DDRB layer

of neurons in the input layer is increased by three times corresponding to the positions: left, center and right.

In fact, for each sign, there are three neurons encoding three different positions in the robot's visual space where the sign might probably be located. Consequently, when one or several signs are detected, their corresponding positions are calculated regardless if they correspond to the expected one to be recognized. Thereafter by means of a competitive mechanism, only the neuron corresponding to the «recognized sign» gets an activity value superior to one while the rest is set to zero and sends the information to the **reflex output position group**. Consequently, the robot can approach the sign by performing the said movement: walking to the left, walking to the right or walking straight ahead.

This layer is composed of the following three neural groups as illustrated in figure 4.24.

Input units

Reflex sign position group: Per each detectable sign known by the robot, there are three neurons encoding a preferred position covering in all, the entire robot's visual field of view. Each neuron encodes a position (x,y) calculated in pixels within the image space and

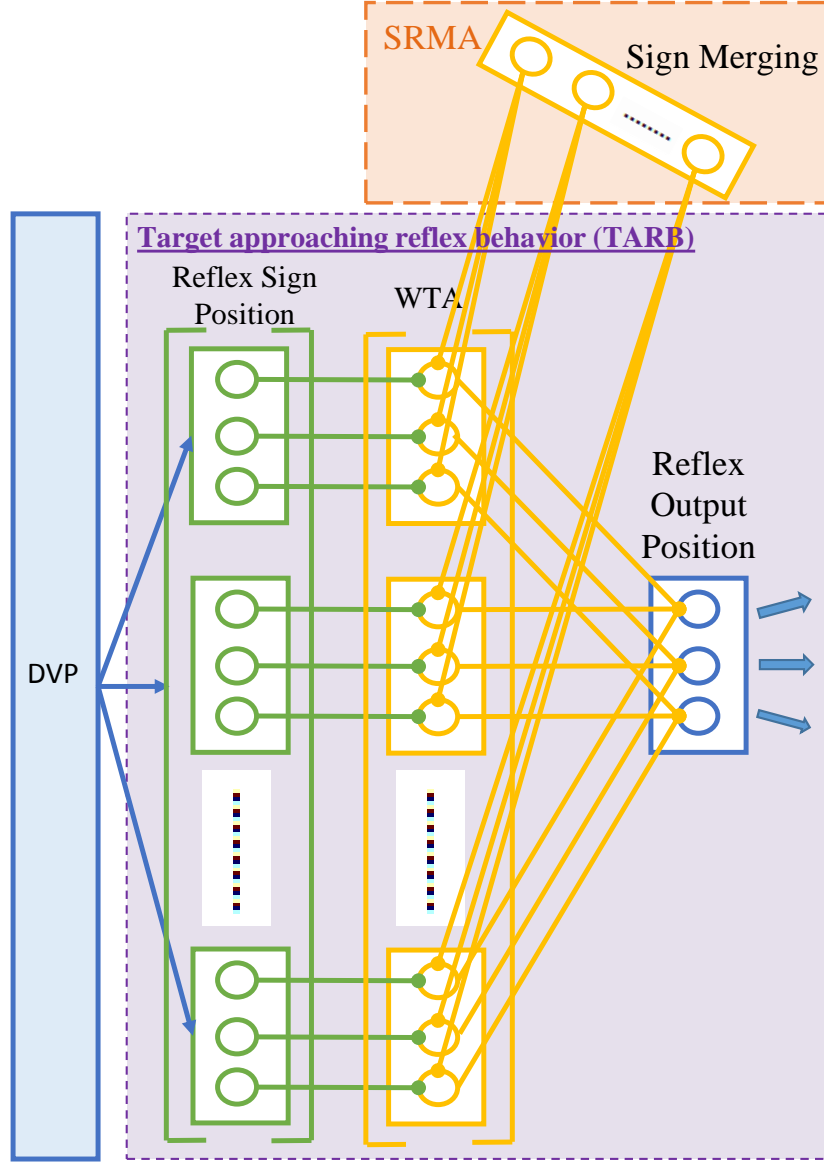


Figure 4.24: Detailed view of the Target Approaching Reflex Behavior layer (TARB)

it is compared to the position of the detected sign calculated in pixels with respect to the referential origin within the same image space. All neurons behave as neural fields which activity can be expressed as a non normalized gaussian activity profile

$$a_j = \exp - \frac{(\alpha - \mu_j(t))^2}{2\sigma^2} \quad (4.3)$$

Where α represents the position j_{th} of the detected sign and μ_j the preferred direction of the neuron j .

Each preferred direction is computed as:

$$\mu_j = \frac{dim}{4} + \frac{d\mu_i(t)}{4} * k, k \in (0, 1, 2) \quad (4.4)$$

Where, *dim* is the dimension of the image given in pixels, and *k* the corresponding number of the neuron per sign (here three neurons per sign). Hence, the same computation is performed for each neuron in the group with the same α value and since their preferred direction is different, only the closest neuron value to α results with the maximum activity value and consequently gets to encode the sign position.

Internal units

WTA group: The competitive mechanism «winner-take-all» enables the neuron with the highest activation value to stay active whereas all the other neurons are set to zero (see equation 4.2). While the *reflex sign position group* sends as input the position of the signs in the robot's visual space, the input coming from the *sign merging group* allows activating only one of the three neurons corresponding to the recognized sign and thus send it to the *reflex output position group* for the movement to be performed.

Output unit

Reflex output position group: It sends the resulting movement of the corresponding activated neuron to the *motor output(MO)*.

4.2.3.3 Layer Convergence

The three layers described above converge towards the *motor output group*, which comprises of six neurons corresponding respectively to six possible movements: turning left, turning right, walking left, walking right, walking straight ahead and turning left as a reflex movement (figure 4.25). The activation of one excludes the others' depending on inhibitory and excitatory signal connections.

Hence, when the *proximity sensor* (robot close to the sign) is activated, the activation values of the *reflex output position group* are inhibited in the *motor output group* and conversely if the robot is far from the sign the direction movements are inhibited.

In the case the movements have been performed by following the reflex movements from the *reflex Output Direction* when looking for the next expected sign, the *reset neuron group* allows inhibiting the activation of the *motor output group* when the *Learn Output Direction* has been triggered by the DDRB and not by the SRMA layer.

The complete architecture is depicted in figure 4.26. For visibility reasons, the names of the neural groups are not exposed. The reader is invited to see each layer presented above for the name details.

4.2.3.4 Layer interaction

The interaction between the layers can be explained by considering different situations within the navigation task. Hence, given a sign sequence, it is necessary to compare it to what it is currently being perceived in the environment to find each expected sign (one after another) in order for the robot to achieve to its final destination.

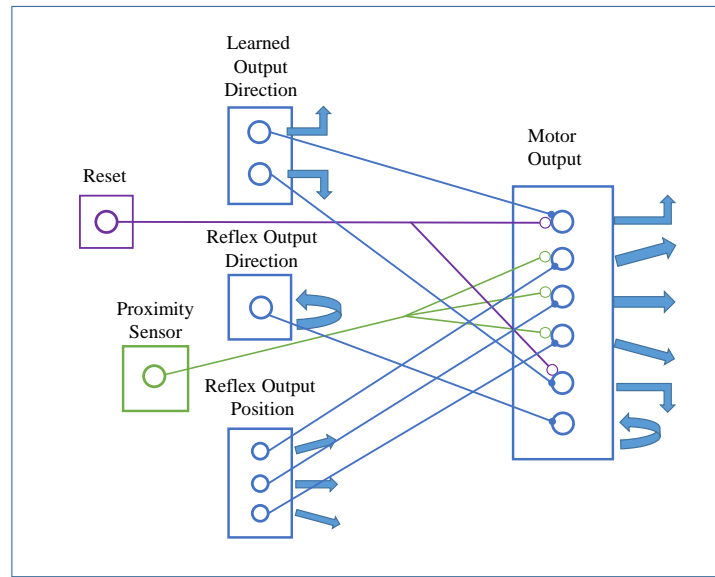


Figure 4.25: Layers convergence towards the Motor Output group.

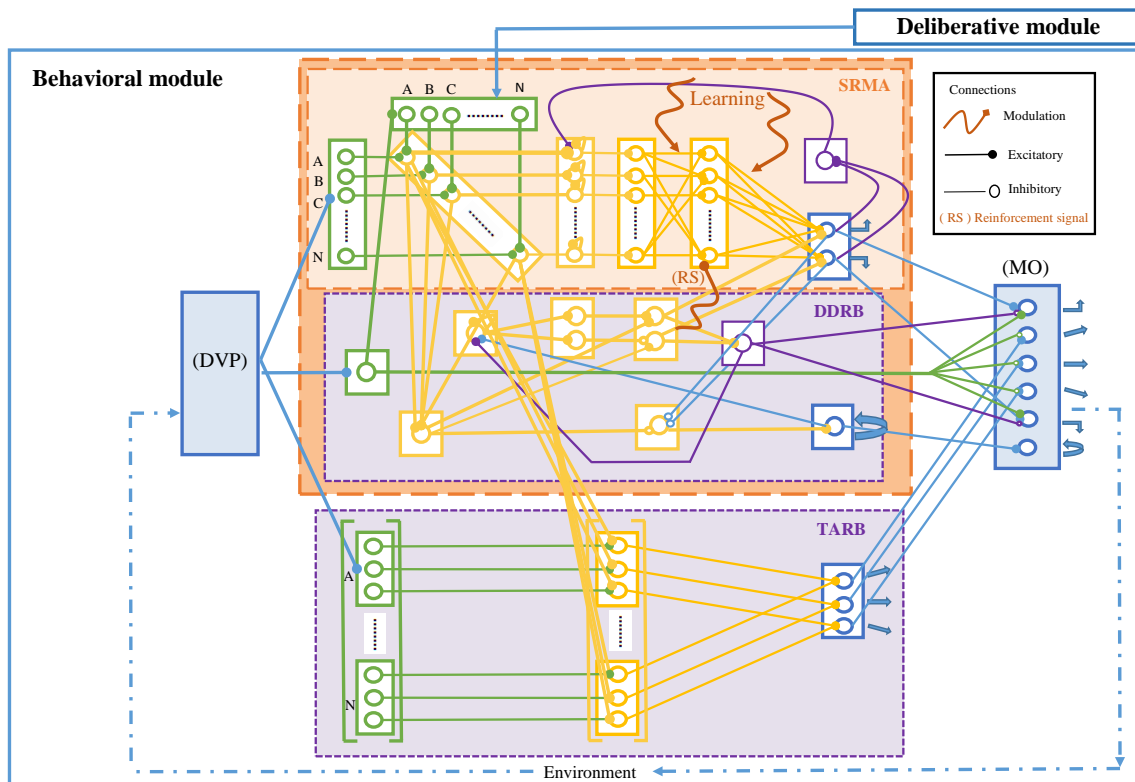


Figure 4.26: Detailed view of the three layers in the behavioral module

The interaction of the layers is next given by describing four different situations concerning a perception and action cycle and covering the whole behavior of the architecture given the simple sign sequence A, B, A, B .

For each situation, there is a single action resulting from the merging of the perception and the *a priori* information and only the neural groups involved by their activation are illustrated:

1. Situation 1: The expected sign is detected, hence recognized, but the robot is far away from it (see figure 4.27).

Initial parameters:

- Sign expected from the sign sequence: A
- Signs detected from the visual perception : A and N
- Locations of the detected signs within the robot's visual field : sign A to the right side and sign N in the middle
- Proximity sensor activation: the robot is far from the sign; therefore, the proximity sensor is not activated.

Functional behavior:

Even though two different signs ("A" and "N") have been detected in the *Sign detection group* of the SRMA layer, only the neuron corresponding to the expected sign "A" given by the sequence is activated in the *sign merging group*. The activation is then propagated to the short-term memory group for further processing. However at this point, little matters if the directional meaning of the sign is known or not to be performed or learned respectively. In fact, since the robot is far away from the sign (*proximity sensor* not activated) it is necessary to first approach the sign and then decide on what movement to perform. This is possible by performing the reflex approaching movements computed by the TARB layer. However, alike to the SRMA layer even though the positions of both detected signs "A" and "N" have been computed in the *detected sign position group* (right and center respectively), only the neural sub-group corresponding to the recognized sign "A" is activated in the WTA group which propagates the activation value to the reflex output position group.

Finally, since the *proximity sensor* is connected via inhibitory links to the neurons allowing the robot to approach the sign in the *Motor Output group*, its inactivity allows to only enable the activation of the neurons corresponding to the movements : some steps ahead towards the left, the right or straight ahead.

2. Situation 2: The expected sign is detected, thereby recognized. The robot is close to the sign and the directional meaning is known (see figure 4.28).

Initial parameters:

- Sign expected from the sign sequence: A

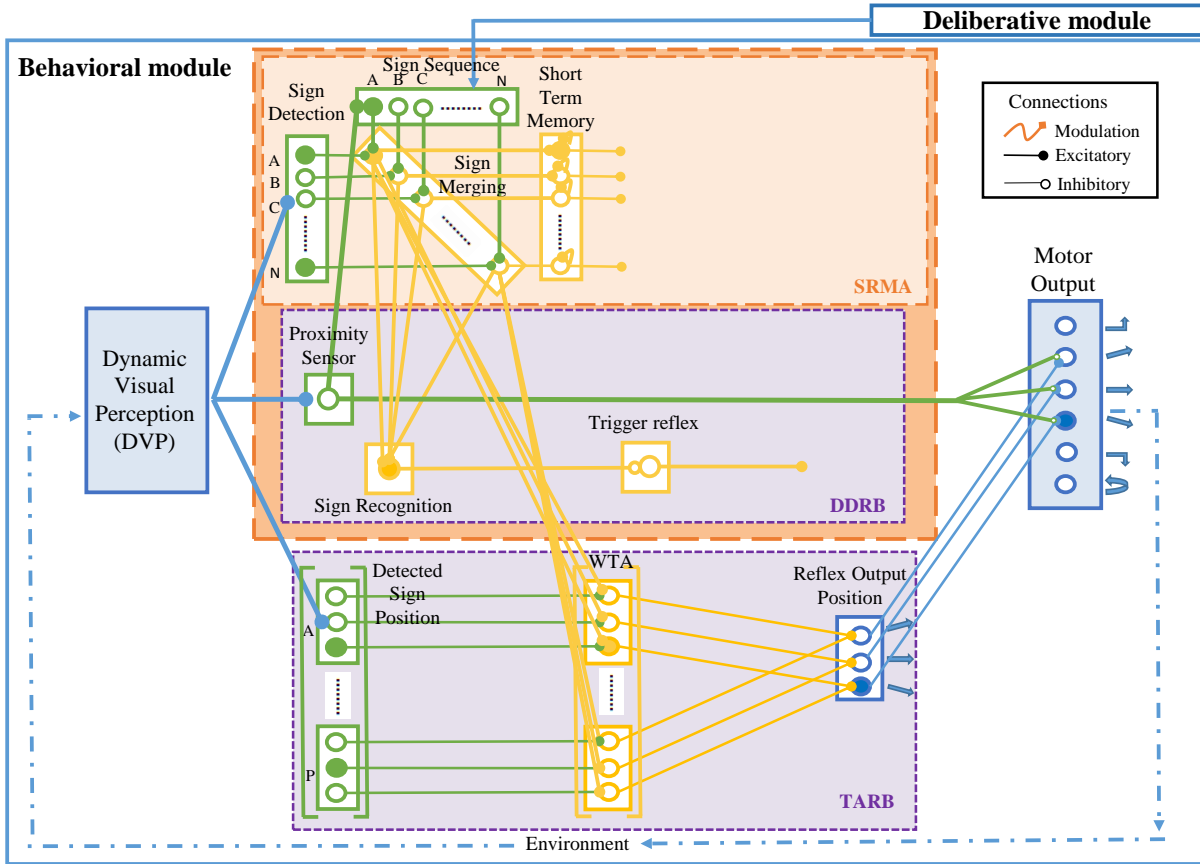


Figure 4.27: Situation 1: The expected sign is detected, hence recognized, but the robot is far away from it.

- Signs detected from the visual perception : A
- Next sign expected from the sign sequence after recognizing sign A : B
- Directional meaning of sign A : is known and is to the right
- Proximity sensor activation: activated. The robot is close to the sign.

Functional behavior:

The information stored by the *short-term memory group* concerning the recognition of sign "A" is propagated to the output of the SRMA layer in order to activate the neuron corresponding to its associated movement and thus, execute any of both possible movements (right movement in this example). Since the *proximity sensor* indicates that the robot is close to the sign, the neurons connected by the excitatory links in the *Motor Output group* are activated. Therefore, only the left and right movements are enabled.

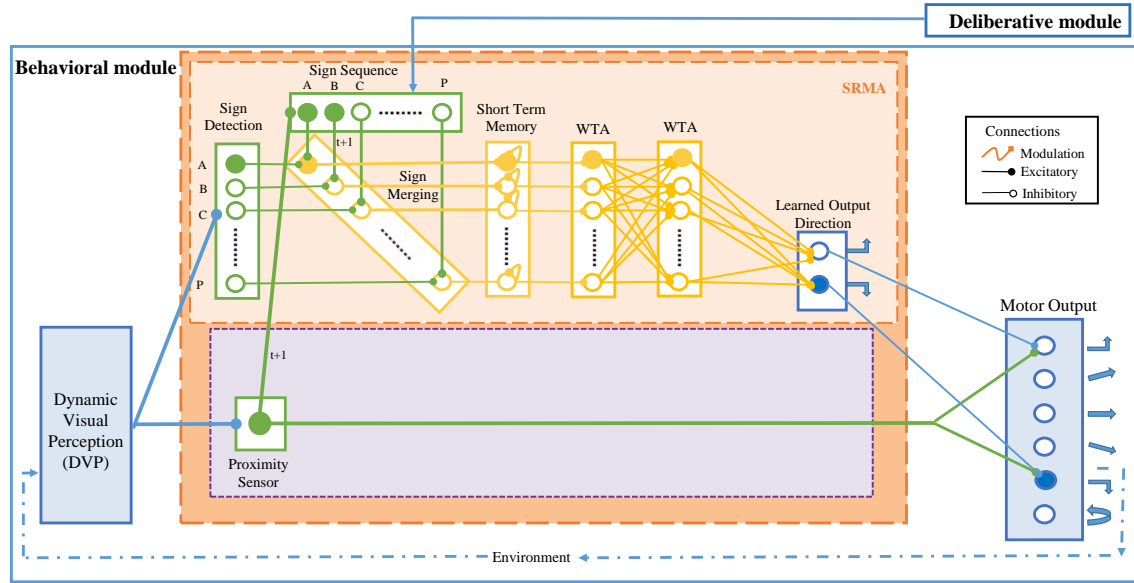


Figure 4.28: Situation 2: The expected sign is detected, thereby recognized. The robot is close to the sign and the directional meaning is known.

Moreover, since there is no longer need of looking for the current sign "A" as it has already been found and achieved, the activation of the proximity sensor triggers the search of the next expected sign in the sign sequence to be looked for in the navigation environment. Thus, the neuron corresponding to sign "B" is activated.

3. Situation 3: The expected sign is detected, thereby recognized. The robot is close to the sign but the directional meaning is unknown(see figure 4.29).

Initial parameters:

- Sign expected from the sign sequence: B
- Signs detected from the visual perception : none
- Directional meaning of sign A : unknown
- Proximity sensor activation: No sign has been detected therefore the sensor cannot give any measurable value.

Functional behavior:

Since the directional meaning of sign "A" is unknown, the robot needs to look in the environment for the next expected sign whose corresponding neuron is already activated in the *sign sequence group* (sign "B"). Hence, the inhibitory connection between the *sign*

recognition group and the *trigger reflex group*, enables the activation of this latter when no sign has been recognized. Consequently, the reflex rotatory movements are performed as long as the expected sign "B" has not detected in the environment. Meanwhile, the *short term memory group* maintains activated the neuron corresponding to sign "A" in order to associate its corresponding movement when sign "B" will be recognized.

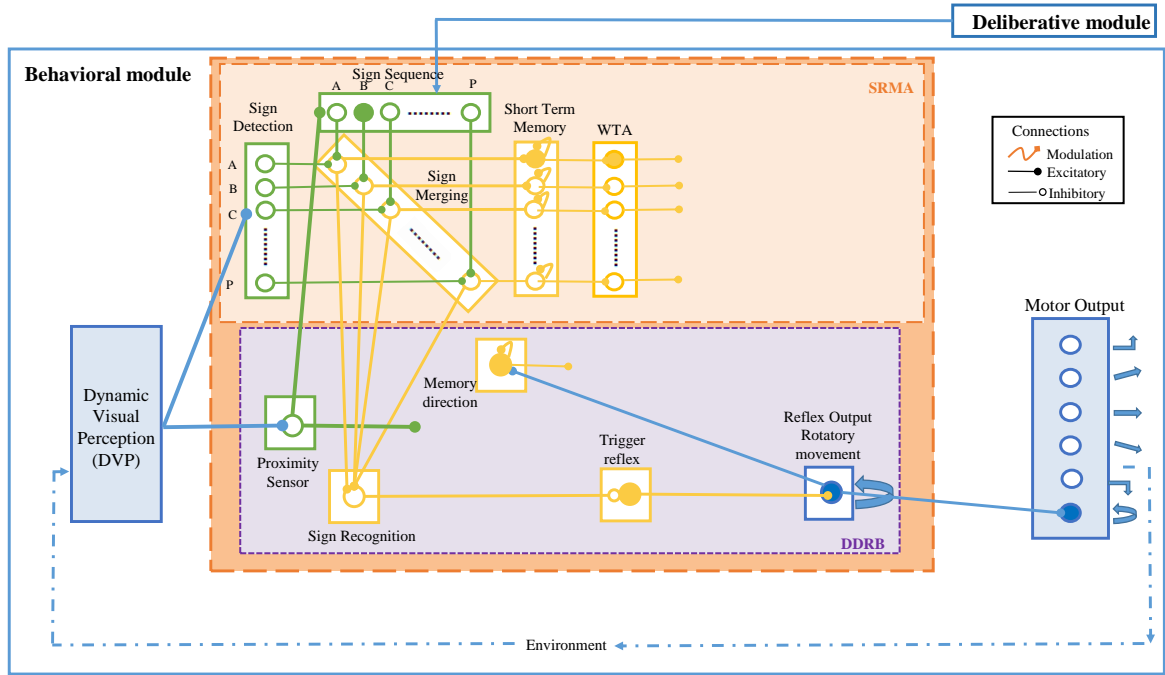


Figure 4.29: Situation 3: The expected sign is detected, thereby recognized. The robot is close to the sign but the directional meaning is unknown

4. Situation 4: The next expected sign is detected, thereby recognized. Therefore, the movement leading the robot from sign A to B has been computed and can be associated to sign A and be learned (see figure 4.30).

Initial parameters:

- Sign expected from the sign sequence: B
- Signs detected from the visual perception : B
- Directional meaning of sign A : To be learned
- Proximity sensor activation: the robot is far from the new sign B; therefore, the proximity sensor is not activated.

Functional behavior:

Once the next expected sign from the sign sequence has been detected, thereby recognized; the movement resulting from the computation of the *memory angle group* in the DDRB layer is learned and associated via a reinforcement signal in the *WTA group* for the SRMA layer. However, since the small reflex rotatory movements have already led the robot to the new sign, the learned movement is not executed in the *Motor Output group*. This movement is prevented by the *reset2 neuron* that sets to zero the values of the *memory angle group* for a new computation. Likewise, the *reset1 neuron* sets to zero all the values of the *short term memory group* so as to allow a new computation concerning the new current sign "B". Simultaneously to the learning task of the sign and movement association, the TARB reflex layer is triggered in order to allow the robot to approach the new current sign ("B") that has been perceived to the right side of the robot's visual field. Situation 1 (previously explained) shows how the robot can approach it.

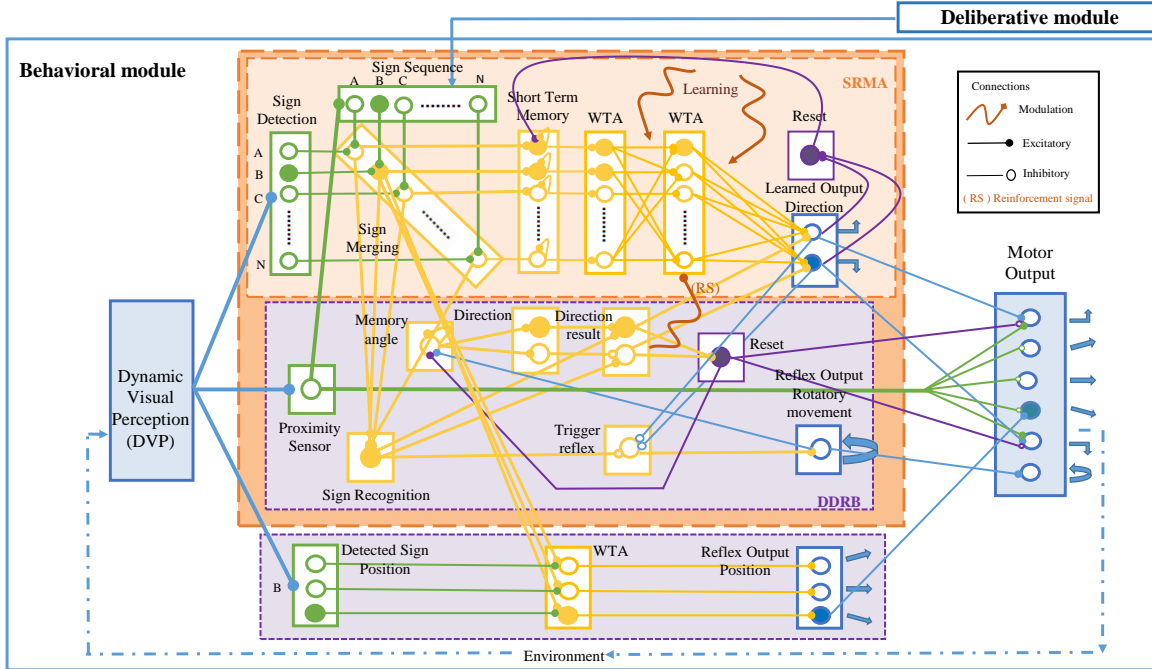


Figure 4.30: Situation 4: The next expected sign is detected, thereby recognized. Therefore, the movement leading the robot from sign A to B has been computed and can be associated with sign A and be learned.

4.3 Experiments in real environment

The efficacy of the proposed architecture was tested with the Aldebaran Robotics' NAO humanoid robot. This platform was found suitable due to its ability to have a rapid visual-perception interaction within a real environment, which is a necessary element to validate the performance of the architecture. The test description below is for didactic purposes, detailing,

confirming and explaining the functioning of the proposed architecture and its salient features at work.

4.3.1 Procedure

The NAO robot can recognize the signs printed in a A4 size paper form using its camera if it is within a range of approximately 0.20 meters to 1.50 meters of distance. Therefore, the navigation environment for the experiment was built so as to always have the next sign to be read within this range. This range can be extended if the printed sign size is bigger.

Then, by placing just two different signs twice in the environment it was possible to verify in one go the real-time learning resulting from a reflex movement, followed by the actual movement triggered by the learned association. The signs were also placed at a certain distance relative to the robot position so as to allow verification of the target approaching reflex behavior. The sign sequence to be followed as illustrated in figure 4.31 was : A, B, A, B.

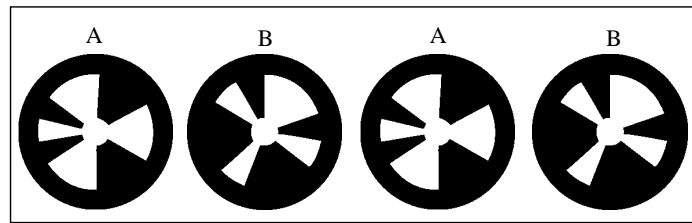


Figure 4.31: Example of signs given in sequence by the user

Thereafter, the robot was placed at the entry of the test environment from which it could distinctly see the first sign that it was expected to recognize, "A". In order to reach the final destination, the robot was supposed to turn right every time it read sign "A" and turn left every time it read sign "B", even though these associations were not known yet to it.

4.3.2 Results

While navigating the environment (see figure 4.32), the robot was successfully able to perform the following intended actions :

- When a sign was detected, it was able to compare it to the corresponding sign from the extracted sequence
- When the comparison gave a negative result i.e. the detected sign did not match the expected sign, the robot ignored the detected sign and continued reflex movements to locate the correct sign
- When the comparison resulted in a positive i.e. the detected sign was indeed the expected sign, and the meaning of the sign was yet unknown, it was able to perform

reflex movements by rotating in one place to search for the next sign and figure out the associated direction and then learnt it

- When the expected sign was not in its visual field, it was able to perform a reflex behavior to search for it
- When a sign was faraway, it was able to get closer
- When a sign appeared again, it was able to recall the learning and perform the associated movement

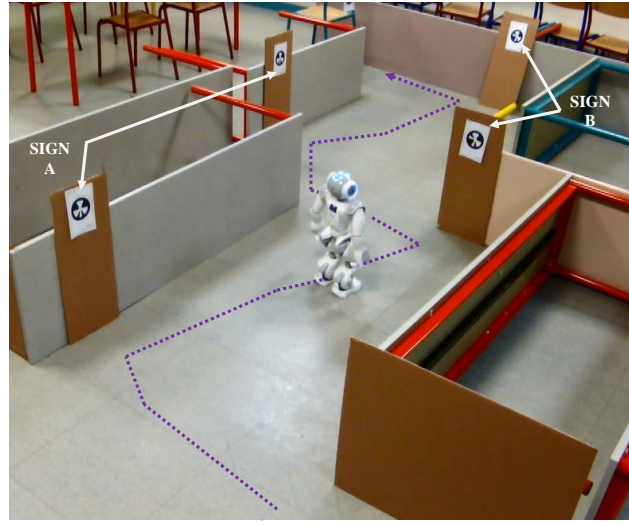


Figure 4.32: Robot navigation within the environment

Figure 4.33 shows a summary of the results obtained in the form of the activation of the output neural groups corresponding to the six possible movements that can be performed by the robot (so far), as well as the activation of the reinforcement signal (**RS**) allowing the association learning, over time. The movements were a result of either the recognition, the proximity or the absence of any signs from the extracted sign sequence. The activities are explained chronologically and refer to the descriptions and figures of section 4.2.3.2. In each of the (a, t) plots shown, (a) is the binary activation of each neural group and (t) the time seconds in terms of a PerAc cycle.

For the sake of simplicity, the plot labeled *Cumulative target approaching movements* combines all movements undertaken in one go by the robot to approach a particular sign using TARB. The three movements are illustrated in figure 4.34.

$t_0 - t_8$ When the robot recognized **sign A** at time **t_0** , its corresponding neuron in the SRMA layer got activated and remained like this while it was still in the robot's field of view (*Dynamic visual perception* plot in figure 4.33). In the meanwhile, it triggered the activation

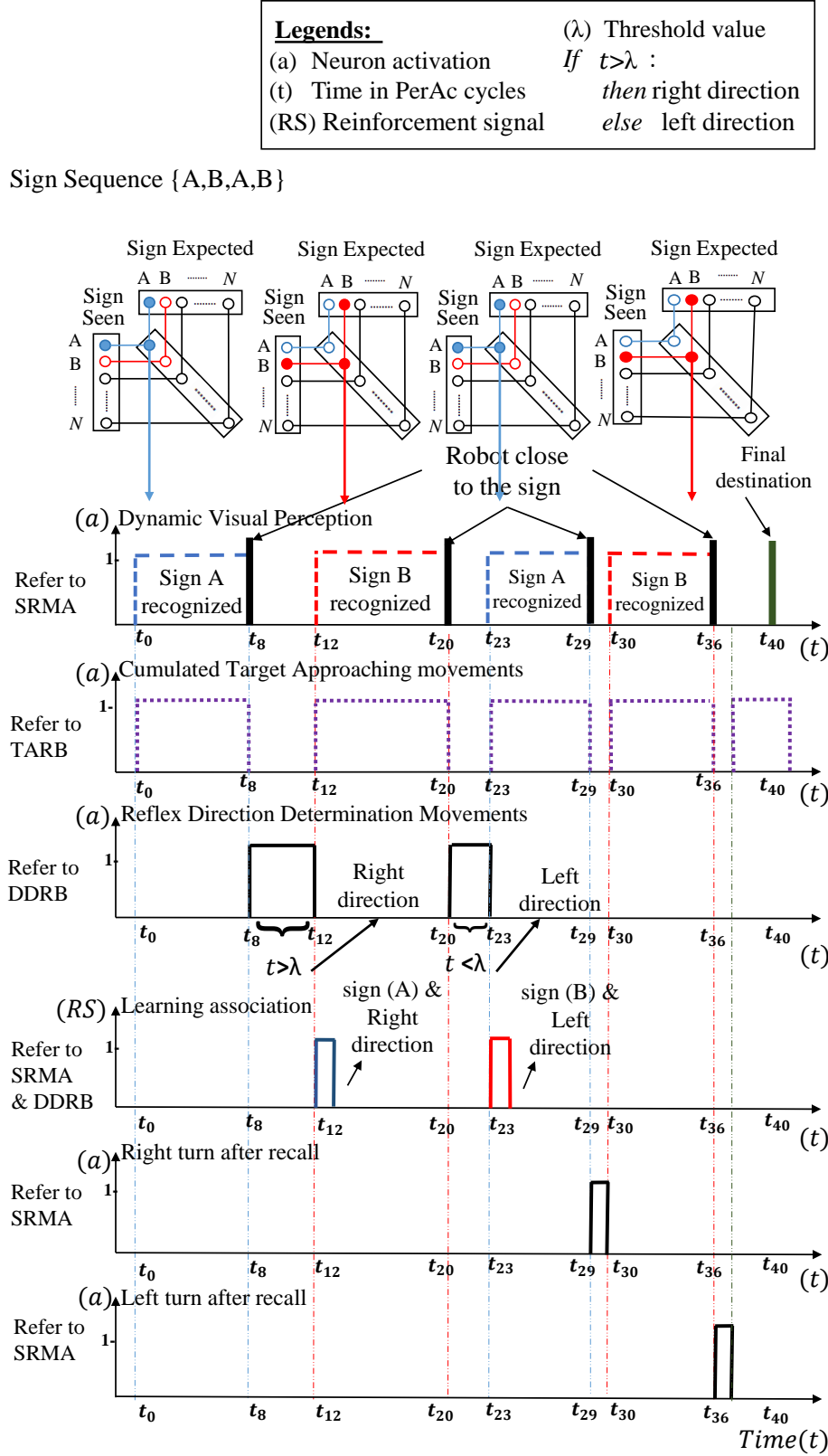


Figure 4.33: Summary of Rhizome 1 results obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal (RS) allowing the association learning, over time. In each of the (a, t) plots shown, (a) is the binary activation of each neural group (t) the time seconds in terms of a PerAc cycle.

of the neurons in the TARB layer allowing the robot to approach the sign(*Cumulative target approaching movements* plot in figure 4.33) until time t_8 . When the robot was close enough to the sign the proximity sensor got activated at time t_8 .

$t_8 - t_{12}$ Since the robot had not associated the current perceived **sign A** with any movement yet, the reflex exploratory action was triggered in the DDRB layer and so was, its corresponding neuron (*Reflex Direction determination Movements* plot in figure 4.33). This continued until **sign B** was detected at time t_{12} .

$t_{12} - t_{20}$ After detection, the total angle of rotation undergone during the reflex movement was computed. The equivalent value (bigger than the threshold value) allowed the association learning of sign **A** with the **right direction** movement triggered by the reinforcement signal **RS** at time t_{12} in SRMA layer(*Learning Association* plot in figure 4.33). While this was taking place, the robot was already performing movements to approach the new sign **B** in the TARB layer(*Cumulative target approaching movements* plot in figure 4.33). The activity of the proximity sensor got a positive value at time t_{20} , once the robot was close enough to the sign.

$t_{20} - t_{23}$ Alike sign **A** earlier, current **sign B**'s associated movements were not known yet. Therefore, the robot used the DDRB layer actions to locate the next sign from the sign sequence **A** (*Reflex Direction determination Movements* plot in figure 4.33) which took place at time t_{23} .

$t_{23} - t_{29}$ This time, the computed angle of rotation was smaller than the threshold value, thereby, resulting in the learning association of sign **B** with the **left direction** movement triggered by the the reinforcement signal **RS** at time t_{23} in the SRMA layer (*Learning Association* plot in figure 4.33). While this was taking place, the robot was already performing movements to approach the new sign **A** (third sign from the sign sequence) in the TARB layer(*Cumulative target approaching movements* plot in figure 4.33). The activity of the proximity sensor got a positive value at time t_{29} .

$t_{29} - t_{30}$ Now that the robot had learned the associated movement of **A**, the SRMA layer allowed it directly performed the **right direction** movement as soon as the proximity sensor was triggered at time t_{29} . The much shorter execution time of this movement on the plot is especially notable, showing the advantage of learning associations.

$t_{30} - t_{36}$ The robot approached **B** at time t_{36} after having performed some target approaching movement directed by the TARB layer (*Cumulative target approaching movements* plot in figure 4.33).

$t_{36} - t_{37}$ The robot executes the associated learnt movement (**left direction**).

$t_{37} - t_{40}$ The robot got to its final destination at time t_{40} after having performed some target approaching movements directed by the TARB layer (*Cumulative target approaching movements* plot in figure 4.33).

As a result, the robot was able to successfully learn the meaning of signs using reflex movements until time $t_0 - t_8$, and thereafter it was able to apply the learning effectively by recalling the movement in a shorter period of time.

Figure 4.34 illustrates the summary of the movements undertaken by the robot to approach the two first signs from the sign sequence by using the TARB layer. It follows the

same reasoning given in the above overall results.

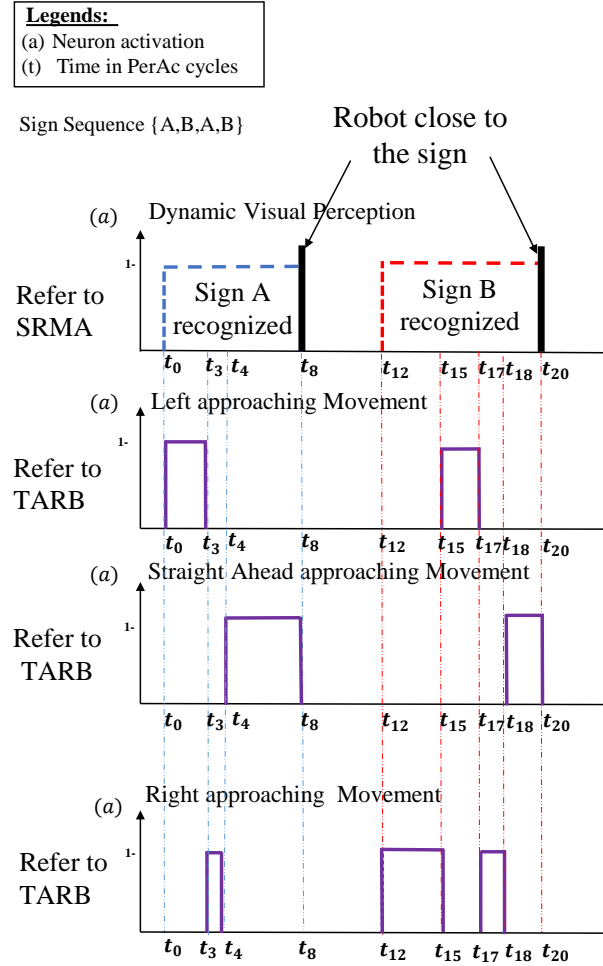


Figure 4.34: Summary of the movements undertaken by the robot to approach a particular sign by using the TARB layer.

4.3.3 Discussion

As in every system, the realization of experiments in real time gives a better insight of the scope and limitations of the proposed system and rises up unexpected problems that were not considered before. It allows easily distinguishing the problems related to the robotic platform from those inherent to the architecture and thus tackle them.

In our case, the overall architecture was conceived by considering in advance the limitations and constraints given by the robotic platform we employed.

For instance, in the absence of a panoramic camera or a camera capable of in-place 360°

degrees rotation, the NAO robot had to perform rotational movements around itself to look for the next sign during its learning phase. Even though speed was not a criterion in the above test, such movements could be inefficient, (although not ineffective). Utilizing head movements, peripheral vision, additional cameras/sensors, or even using an alternate platform instead, may increase the speed of task completion if speed is a requirement in the navigation task.

As for the problems inherent to the architecture, even though the behaviors performed by the robot within the test environment were as expected during the experimental phase, an unforeseen problem arose when the same experiments were carried out in different environments.

In fact, it was noticed that while the robot was approaching a given sign, it would sometimes lose the track of it by failing to perceive it within its field of view. This problem led the robot to perform the rotational movement triggered by the DDRB layer in order to look for the same sign. Such reflex behavior could have been the logic solution to do if it had not been for the fact that after having found the sign again, the learning signal was activated and a directional meaning was directly associated to the sign.

However, at that point the problem was not significant and did not cause any trouble for the robot to continue its way. The actual problem became then noticeable when the same sign had further appeared and the robot had to recall the associated directional movement of the current sign. It appeared that the robot had associated two different, and most of the time, opposite movements to the same sign and when recalling the movement, it performed one after the other.

The source of the problem happened to be the lighting variation to which, cameras are usually quite sensitive and to which unfortunately we had failed to consider among the parameters of the given scenario.

A temporary solution was taken in order to prove the expected behaviors previously hypothesized and already described in the result section. It consisted of repositioning and tilting the sign in a way that it would be well lit whenever the robot lost track of it. However, this problem needed to be taken care of in a more permanent way.

Hence, in order to overcome this problem some neural groups were added to the architecture. The reader can refer to the appendix section for a detailed description of these groups.

The complete architecture as illustrated in the introduction is depicted in figure A.7 with the new ***Lost Sign Searching Reflex layer***:

4.4 Conclusion

A hybrid neural-based architecture, Rhizome 1, enabling autonomous navigation based on both *a priori* information at the start of the journey, and dynamic information from the immediate environment during it, has been presented in this chapter as an alternative hybrid control architecture for mobile robot navigation. The *a priori* information is provided by a program that has computed previously the navigation path and the robot is able to

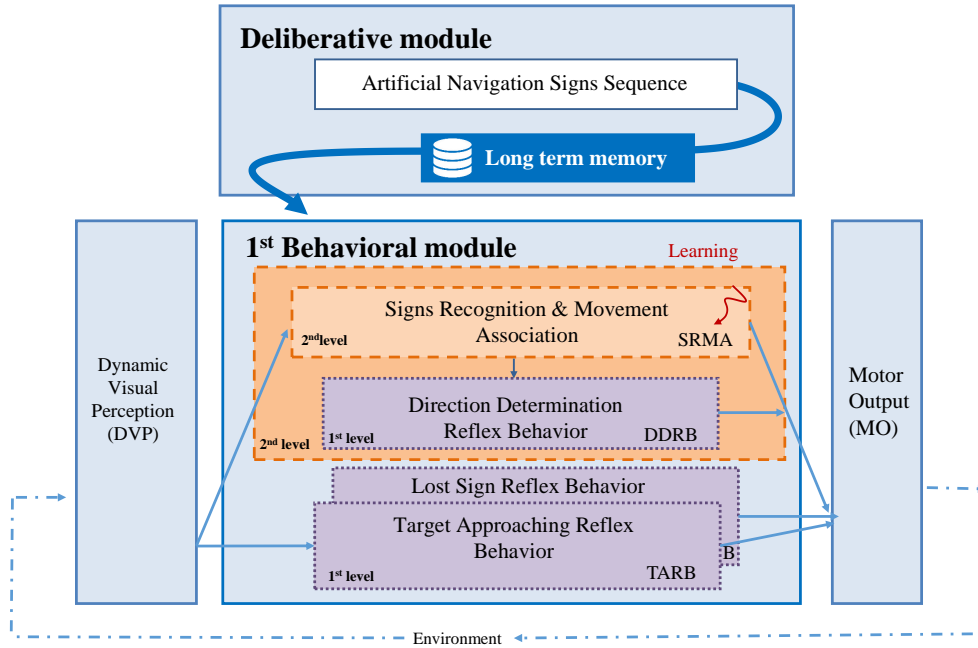


Figure 4.35: General view of the complete Architecture. Rhizome 1 is composed of two modules, one deliberative and one behavioral. This latter is composed of three reflex behaviors layers and one recognition layer

integrate it, to compare it to the dynamic visual information and to reach a specific place while performing online learning of sensory-motor associations. Experimental results obtained from the physical implementation of the architecture in an indoor environment have shown feasibility of this approach.

Furthermore, thanks to its generic composition, the proposed architecture has shown to be easily adaptable to new behaviors and tasks. Indeed, it is possible to develop the architecture further with respect to robustness and completeness by simply adding new layers without modifying the already in-built components or layers. For instance, in the discussion section it was shown that by adding new neural groups to the different layers, the robot could overcome a problem that came up while navigating the environment.

Similarly, another layer which, is omitted in this thesis for the sake of brevity in favor of detailing the more relevant contributions, has also been added to the architecture. This layer allows the robot to learn the association between the current sign and the movement that led the robot to it from the previous visited sign. It is called the «**retour-au-nid**» layer (nest returning in English) and is based on the biological mechanism used by some insects like bees, ants and wasps [Wehner and raber (1979)] , [Cartwright and Collet, 1983] that allows them to return to their home from wherever position they are at. The reader may refer to the state-of-the-art section 2.2 and 2.2.4.1 of chapter 2 for a deeper insight of such mechanism.

As a result, the robot not only is able to navigate towards its final destination, but also,

it is able to come back to any of the visited places as well as the starting point from any other position.

In the next chapter, we present the Rhizome 2 architecture where the robot is able to extract the sequence of signs by itself from a paper-based map, instead of actually waiting for a person or program to provide it the *a priori* information. Moreover, the additional information given by the map generates the addition of new neural groups as well as the modification of some neural links.

RHIZOME 2: Autonomous map-based robot navigation

Contents

5.1	General Description	131
5.2	Implementation- Rhizome 2 Architecture	132
5.2.1	Overall description	132
5.2.2	Deliberative module – Floor plan analysis	135
5.2.3	Behavioral module - Neural structure	145
5.3	Experiments in real environment	146
5.3.1	Procedure	147
5.3.2	Results	148
5.3.3	Discussions	156
5.4	Conclusion	156

5.1 General Description

Digital floor plans of buildings (such as hospitals, schools, residential complexes or factories) are typically not as readily available as digital maps of entire cities, countries etc. Hence, when trying to navigate an unknown building, one has to rely on floor plans available in physical form at the entrance of the building or on paper to achieve one's final destination. Since such a floor plan provides one of the fastest way to access comprehensive information about the inside of the building, it can be used as a priori information for reference purposes in the navigation task of a mobile robot.

Consequently, Rhizome 2 has been conceived in order to combine the information extracted from a paper-based floor plan with the dynamic visual information. This is achieved by having the robot recognize some navigation signs from the floor plan and then look for them in their expected sequence in the environment.

The global knowledge of the world is represented by the paper-based floor plan that is placed in front of the robot's camera just once, before the navigation activity starts (see figure 5.1 (a)). It contains the important information to define a potential navigation trajectory in a deterministic scenario. In this work, this information is represented by navigation signs used as reference points that are expected to be seen by the robot in the real world navigation (same black circles with specific white patterns drawn within seen in chapter 4,

but associated with left and right directions as shown in figure 5.1(b). By means of computer vision methods the robot «reads» the floor plan, generates an optimal plan to reach the goal, recognizes the navigation signs within the path, computes the directional meaning each sign denotes (turn right or turn left), extracts and stores the sequence of signs arranged from the closest point to the furthest with respect to the starting point.

The robot then integrates the extracted sequence of signs together with the directional signs meaning into a neural system and by comparing it with the dynamic visual perception of the environment in real-time (see figure 5.1 (c)) it is able to recognize the signs and perform the movement associated to their directional meaning.

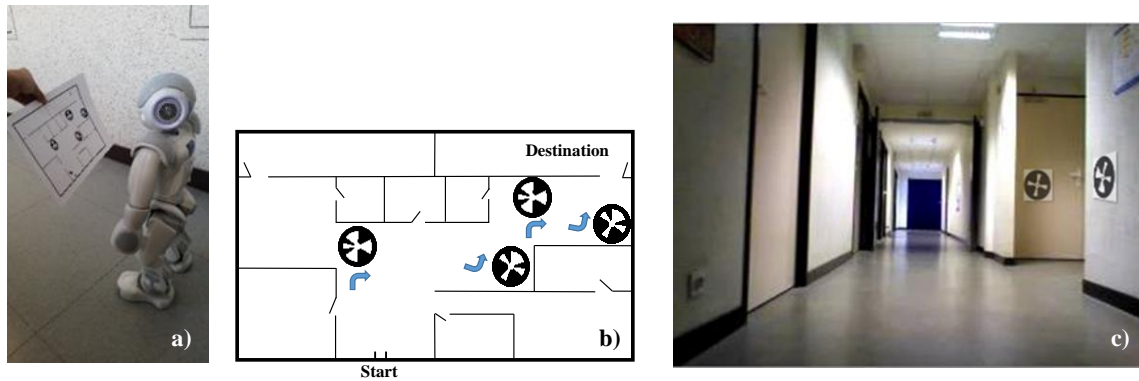


Figure 5.1: (a) Nao robot reading the map. (b) Map with the signs and their corresponding directional meaning. (c) View of the environment with the navigation signs in real time.

Hence, the same functional behavior of the navigation process explained in chapter 4 applies to the description of the navigation process of Rhizome 2, with the exception that once the robot is close to the recognized sign it can directly turn towards its corresponding direction which was previously computed by analysing the map (see figure 5.2).

5.2 Implementation- Rhizome 2 Architecture

5.2.1 Overall description

The overall architecture integrates the information provided by the floor plan analysis into two organized neural structures. Hence, it is composed of three modules as illustrated in figure 5.3.

A deliberative module, where a thorough analysis process allowing the extraction of the floor plan information takes place and **two behavioral modules** that integrate the resulting information and which by the use of a cognitive mechanism, perform recognition of a particular sign and learning of its association with its corresponding directional meaning.

On one hand, **the deliberative module** is composed of a floor plan analysis system.

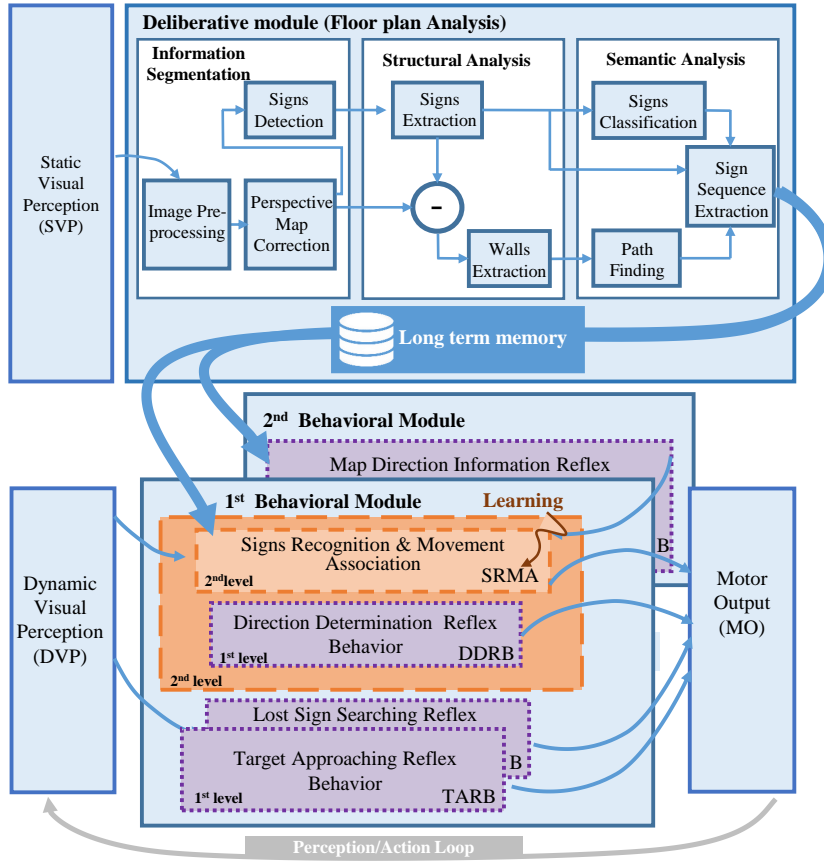


Figure 5.3: Overall Rhizome 2 architecture composed of three modules: One deliberative composed of a floor plan analysis system and two behavioral modules integrating the information from the deliberative module and allowing the robot navigation.

sign denotes is incorporated in the **second behavioral module** which is also based on a perception-action mechanism [Gaussier 1995].

The **second behavioral module** is directly connected to the learned *output direction group* of the **first behavioral module** and by activating a reinforcement signal, the association between the expected sign and its directional meaning is learned. Additionally, as this module is activated, the *direction determination reflex behavior layer* of the **first behavioral module** is inhibited by an inhibitory link which connects both modules. Thus, the exploratory directional reflex movement is no longer needed (figure 5.7).

Hence, if we take the functional diagram presented in the introduction, each module intervenes on every action as illustrated in figure 5.8.

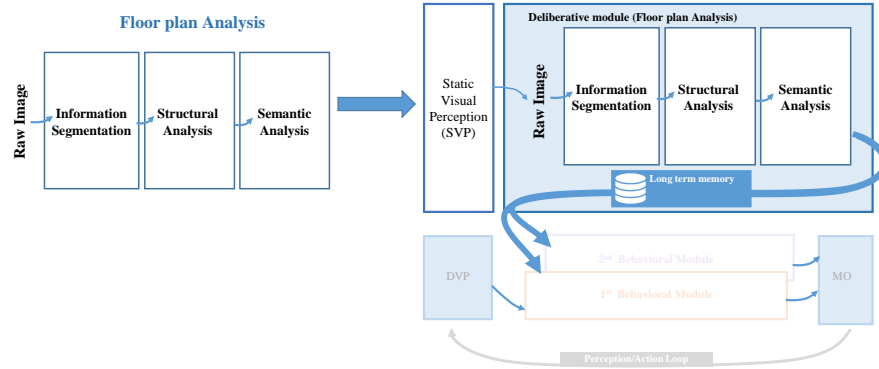


Figure 5.4: State-of-the-art floor plan analysis workflow (left) used in the deliberative module of Rhizome 2 (right).

5.2.2 Deliberative module – Floor plan analysis

5.2.2.1 Overview

When working in real time situations, reading and analyzing a document from an image acquired from a mobile device such that of a robot is a challenging task. It not only has to be accurate but it also has to be fast so that the robot can react accordingly. Image acquisition of documents under constrained and dynamic conditions may cause not only degradation but also distortions on the captured image itself due to unpredictable orientation, distance, lighting conditions etc.

Different scenarios might be possible. For instance in the case of this work, since there is no use of a planar support when acquiring the image, the map might not be completely or at all within the image frame. It might also be too small or suffer from perspective distortion if the image captured is not orthogonal to the plane of the camera. All these circumstances make it hard to properly identify and retrieve the content of the map. Therefore, a set of processes after image acquisition has been implemented allowing the analysis of the floor plan in order to permit the robot to extract the relevant information for the navigation task. This process is achieved through three main stages of floor plan analysis as illustrated in figure 5.9.

The **information segmentation** process identifies and separates different types of information. First, in order to remove any insignificant elements that may cause problems when analyzing the image a pre-processing is applied. Then, a perspective correction is used in order to easily distinguish the important information since the acquisition of the images is not done on a planar surface.

Thereafter, the **structural analysis** is in charge of extracting the information within the map separately. After detection of the signs in the last stage, the regions around the signs are subtracted from the images. As a result, only the walls remain in the image.

Finally, the **semantic analysis**, which consists of two different process that are simul-

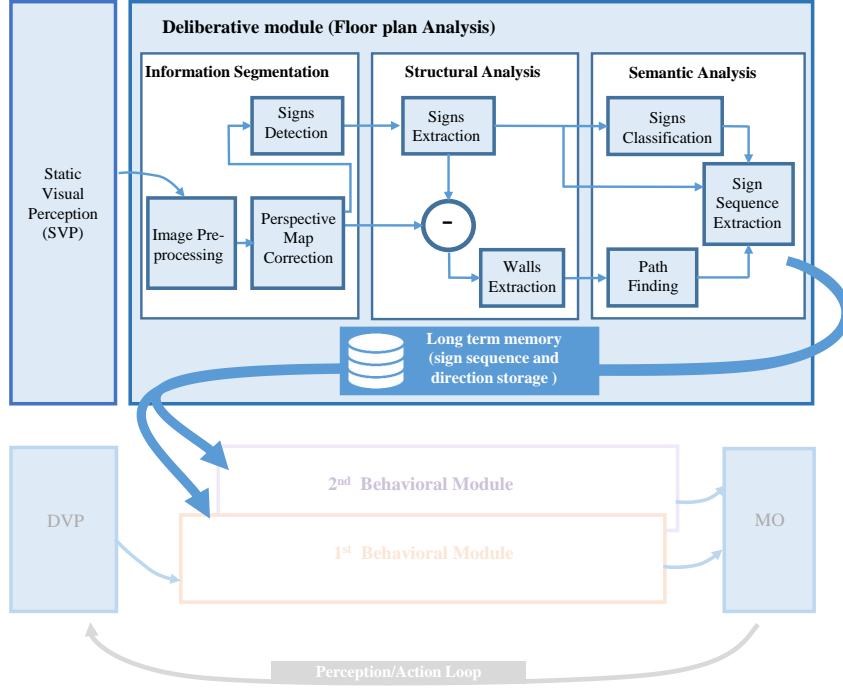


Figure 5.5: Deliberative module composed of a floor plan analysis system. It consists of an information segmentation process, structural analysis process and semantic analysis process. The resulting information (sign sequence and sign directional meaning) is stored in the long term memory.

taneously applied to the resulting images from the last step. On one hand, a classification process allowing to recognize the signs extracted previously and on the other hand, the computation of the path leading to the final destination from a starting point.

As a result, the robot is able to compute the path leading towards the final destination and thus extract the sign sequence based on the computation of the path and the information of the signs

5.2.2.2 Information Segmentation

Image pre-processing As the outcome of the segmentation process directly affects the performance of further processing, it is important to find a suitable *image pre-processing algorithm* that allows removal of any noise and help to simplify the representation of the image for better analysis.

A global thresholding has been applied to the images in this work, as it has proven to be the simplest of image segmentation methods while providing a fast and convenient way to perform a good result when working in real time situations [Singh 2010]. However, non-uniform illumination of the document may cause inaccuracies or complete failure of

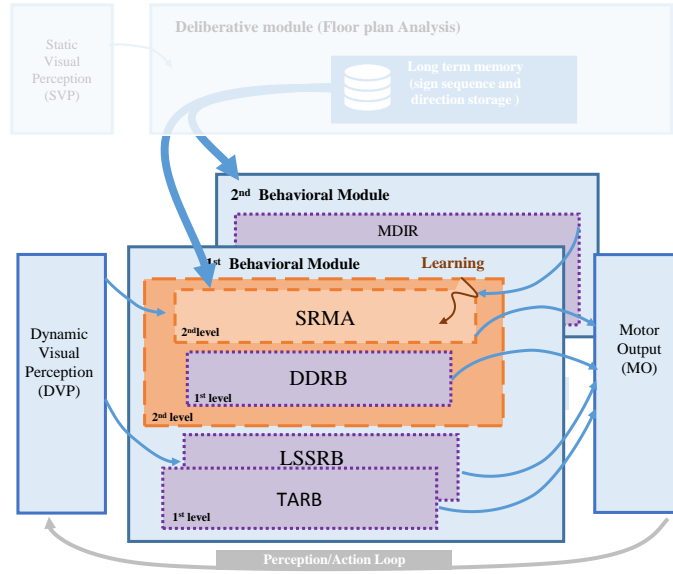


Figure 5.6: Two behavioral modules integrating the sign sequence information and the directional meaning each sign represents.

segmentation. In order to overcome these effects, a black top-hat morphological operator with square mask is first applied on the gray-scale image. This process consists of removing noise and small objects from the image that are not relevant to the analysis of the image. Hence, it improves the clarity of the image outcome and consequently, it makes possible the identification of relevant regions (map) and objects of interest (navigation signs and walls) in the image.

The identification and then extraction of the map from the image is made possible using analysis of the connected components. Therefore, after the thresholding process, two filters are performed based on the contours of the image components in order to select a clear image of the map to make extraction of navigation signs possible. The maps used in this work are considered to be printed in A4 format and all the threshold values are calculated based on the values retrieved from a potentially good image previously used as reference.

The retrieval of the contours is possible by following the borders of the connected components using the algorithm proposed by [Suzuki 1985]. Each contour is stored as a vector of points and they can be organized by following a hierarchy order. Since the aim of this step is to separate the map from the background, only the outer contours corresponding to the boundary map are retrieved and analyzed.

The application of these filters based on the size of the connected regions permit rejection of images. Hence, the first filter is based on the length of the contours, more particularly, on the size of the output vector of points, which is, compared to two thresholds values (one minimum and one maximum previously calculated). If the size of one contour is inferior to

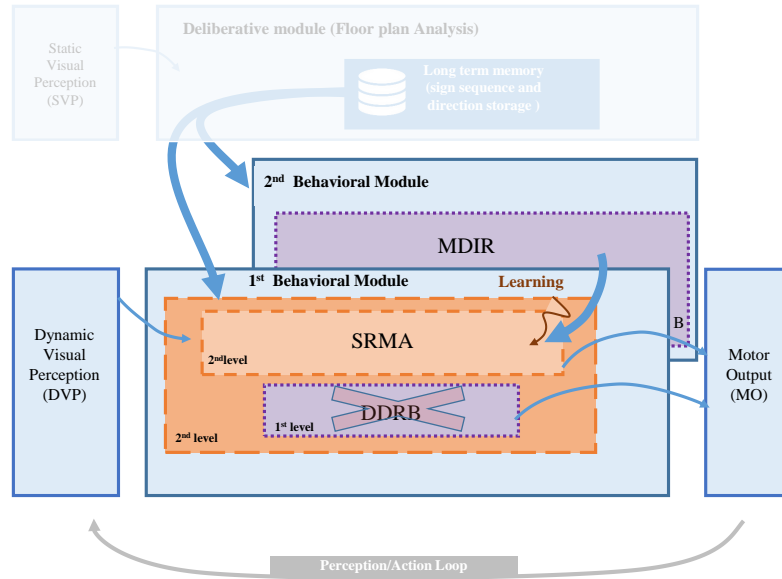


Figure 5.7: Functioning of the two behavioral modules. Since the second behavioral module integrates the sign directional meaning, the Direction Determination Reflex Behavior (DDRBR) layer of the first behavioral module is no longer needed, therefore it is inhibited.

the minimum value, it is likely to correspond to irrelevant components outside the convex hull in the background or to an incomplete map boundary (map outside the camera frame). If conversely, it is superior to the maximum value, something must probably be intersecting the contour (i.e. user fingers holding the map when acquiring the image) and, thus, a part of the internal contours is also detected. Therefore, only the vectors whose size is between these two values, has a much higher probability of being the complete map boundary.

Once the image has passed through the first filter and some contours have been selected as potential map boundary contour, a second filter takes place. Indeed, a bounding box is created around the potential map boundary contour and its area is compared to a threshold value. Only the area value superior to the threshold value is selected and the map edges can be identified.

Hence, only the images containing the complete and visible map are selected for further processing.

Perspective map correction Most of the methods proposed in the literature for perspective distortion removal are based on page layout and document content such as text lines and vertical paragraph margin (VPM) [[Dance 2001], paragraph formatting [Clark 2003] and stroke boundaries and tip points [Lu 2005]. The first two methods find horizontal and vertical vanishing points and the last one uses multiple fuzzy sets and morphological operators. Nevertheless, textual information such as the name of the rooms (likely to exist in floor plan

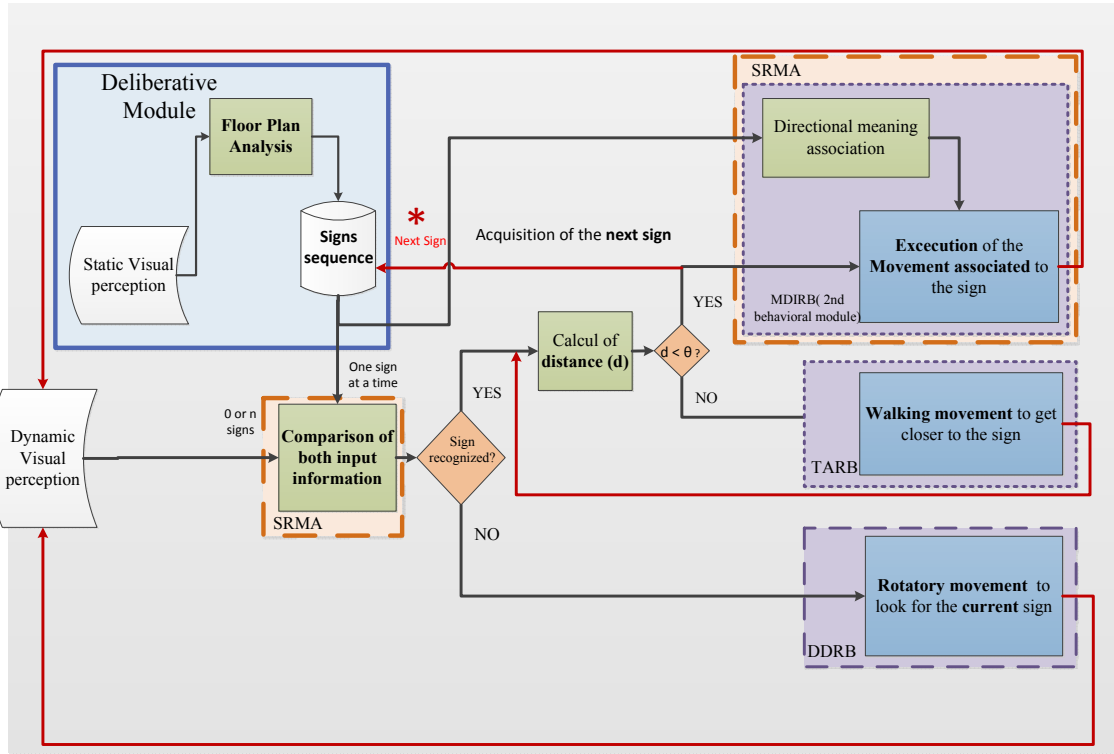


Figure 5.8: Functional diagram of the navigation process behavior allowed by Rhizome 2 with the architecture layers acting on each functional decision.

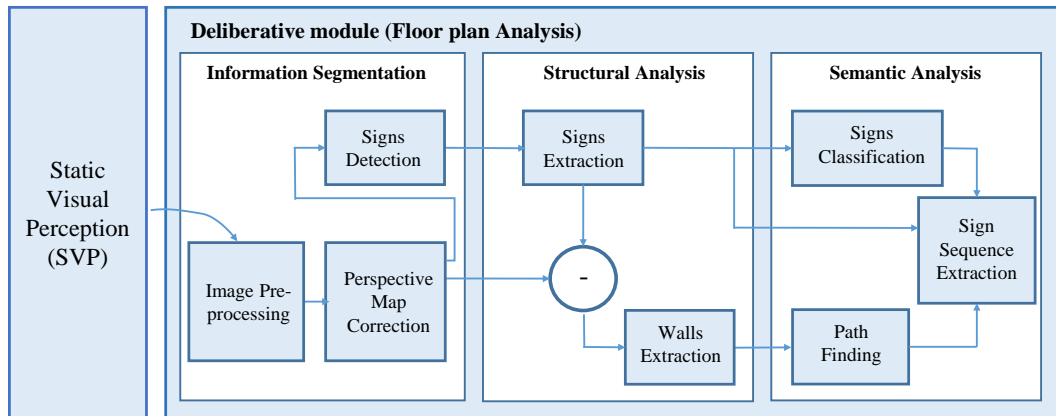


Figure 5.9: floor plan analysis process achieved through three main stages: Information segmentation, structural analysis , semantic analysis based on [Ahmed 2011].

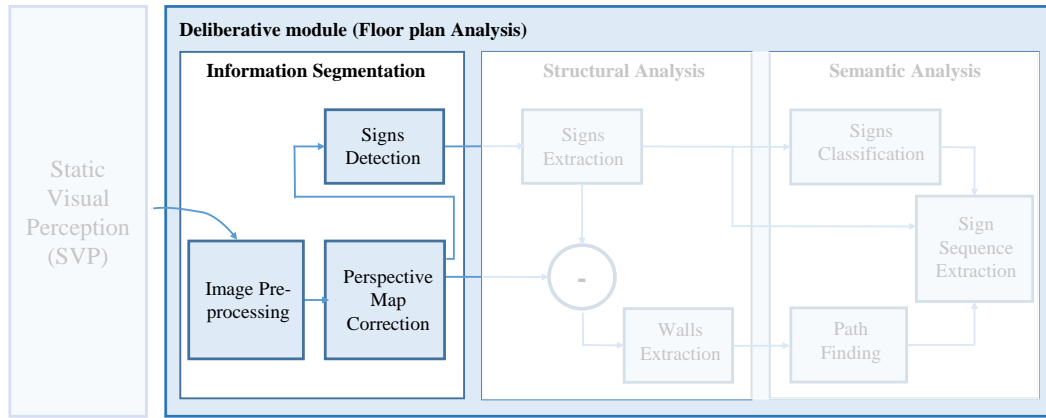


Figure 5.10: Information Segmentation process.

documents) is not exploited in this work. Hence, all the methods based on text, paragraphs or characters are here excluded.

On the contrary, a high contrast document boundary (HDB) like method is used in this work. It finds the corners of the quadrilateral formed by the boundary between the background and the document to transform it into a perfect rectangle as proposed by [Clark 2002] and [Rodríguez-Piñeiro 2011].

Then, detecting the corners directly from the map with a corner detector such as the Harris corner detector would seem to be the simplest way. However, as it detects all the possible corners within the map, the computing time required to select only the outer ones would be high and unnecessary for our task.

Therefore, the method proposed in this work consists of first creating a bounding box around the image, then retrieving the corners coordinates (X,Y) of the bounding box formed to use them as reference points to find the coordinates (x,y) of the corners of the map contour. By calculating the minimum distance between the coordinates (X,Y) of each corner of the bounding box and each vector point of the map contour, the closest vector point to each corner of the bounding box is found.

As a result, the corners (x,y) of the map contour are found. Then the perspective correction can be performed by using a 3x3 transformation matrix calculated with the coordinates of quadrilateral vertices of the map and those of the destination image.

Sign detection The navigation signs detection task can be considered as part of a symbol recognition problem in the area of pattern recognition. In general, symbols can be defined as graphical components which are found in different types of documents and are meaningful in a specific domain. They can be differentiated by their visual properties such as their lines segments, their shape, their gray levels, etc. Even though, the majority of symbol recognition methods assume that symbols have been previously segmented. Segmentation still remains a difficult task as it is not always possible to partition the image into unique constituent

components. However, few methods base their symbol segmentation on features such as connected components, loops, color layers, long lines, etc.

The navigation signs are represented in this work by circle-shape symbols containing specific patterns that allow them to be differentiated from one another. Hence, in order to distinguish them from the floor plan, sign detection is achieved using a feature based approach by using the Hough circles transform [Ballard 1981].

5.2.2.3 Structural analysis

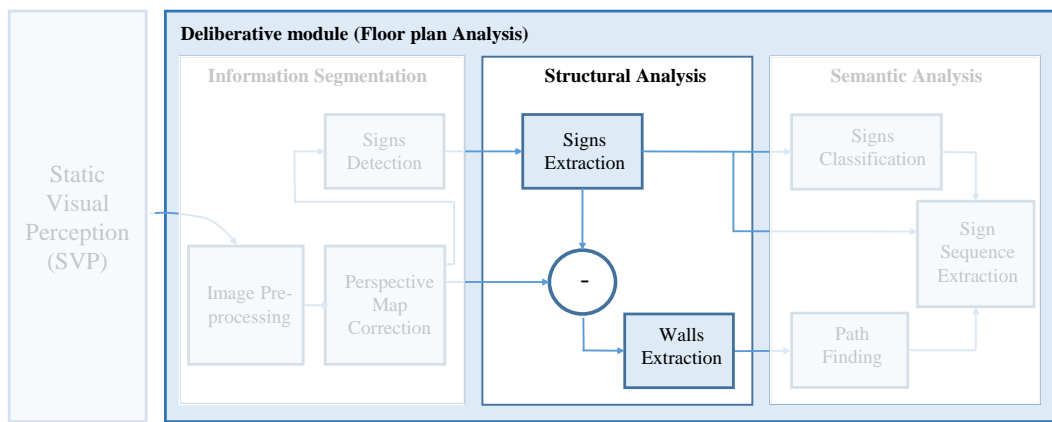


Figure 5.11: Structural Analysis process.

Signs and walls extraction In order to find the correct sequence of signs for navigation purposes, it is essential for the robot to take into account the internal walls as they represent static obstacles that have to be avoided.

Even though this information must be considered all together, information is segmented, processed separately and then merged together. This segmentation process allows avoidance of incorrect results as some information, which might not be required for a given step, might be considered as noise.

The *signs/walls segmentation* process, analyses the map image and converts it into two images. The first one containing only the navigation signs and the second one containing the walls.

The problem of wall detection and separation from graphical information has been solved in [Dosch 2000] by using a morphological filter that separates the image into two images i.e. a thick line image containing the walls and a thin line image containing the symbols. This same method has been enhanced in [Ahmed 2011] by adding a third kind of lines i.e. medium lines in order to retrieve also the outer walls. In [Macé 2010], the authors proposed a method that consists of first detecting the lines that are likely to be the contour of the walls (based on the coupling of Hough Transform with image vectorization), and then verifying the

texture between two aligned lines. Even though these methods seem to be robust, the symbols considered for the application domain correspond mainly to windows and doors related to more complex floor plans.

As previously mentioned, the aim being that of proving a navigation task by means of navigation signs on simple floor plans. Therefore, these approaches are not here exploited in this work. However, the author considers them relevant for future stages of this project.

Hence, the information obtained in the previous step is used in order to extract each sign from the floor plan. By considering the parameters referring to the center position and radius (y_{center} , x_{center} , r) of the circles, denoting the signs, obtained by using the Hough circles transform, it is possible to subtract the information within each circle from the floor plan and copy each of the signs into a new image.

As a result, a «walls image» composed of only internal and external walls is produced as well as a set of «navigation sign images» each consisting of a unique sign.

5.2.2.4 Semantic Analysis

In order to extract the correct sequence of signs from the starting point to the final destination, two different process are respectively applied to each of the resulting images from the previous step i.e. «navigation signs images» and «walls image»: A **sign classification process** to the former and a **path planning computation** to the latter.

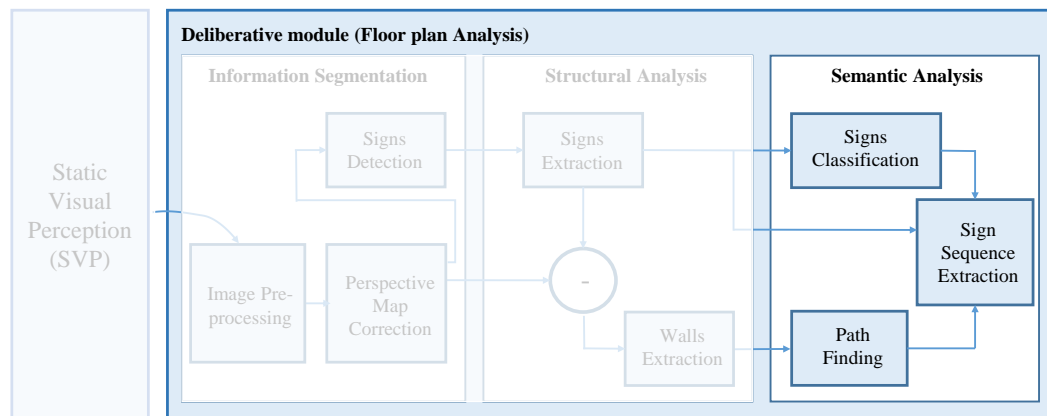


Figure 5.12: Semantic Analysis process.

Signs Classification As part of the first process, it is necessary to recognize the sign images among all possible signs used in this work. In other words, it is necessary to identify the category to which the sign belongs which is commonly known as the classification problem. There are two notable methods available for the recognition of signs in maps or paper-based documents using image processing: keypoints based [Rusiñol 2010], [Rusiñol 2013] or template matching based [Weber 2012]. The former aims to extract some points of interest

(generally the corners) and describe them using some radio-metric features. The latter aims to find small parts of an image that match a template image from existing database. This latter simpler approach was used in this work since a database of the sign images was already available and could easily be recognized by the robot in the environment.

Path planning The problem of finding the optimal path is solved by using a neural implementation of the resistive grid technique (also known as the Laplacian path planning method [Connolly 1990]) [Bugmann 1995]. The neural network comprises a *neuro-resistive grid* in the upper layer and a *spatial memory* in the lower layer (see figure 5.13). The image of the walls extracted during the *image processing* step described earlier is used here as the *resistive grid* where each pixel in the image (or a «node» in the grid) is represented by a neuron in the upper layer, which is connected to its m closest neighbors and one neuron from the lower *spatial-memory layer*. The *spatial memory layer* stores the information of the target (final destination) and obstacle positions given by the image, and is used to constrain the activity of the corresponding node in the *resistive grid* (upper layer neuron).

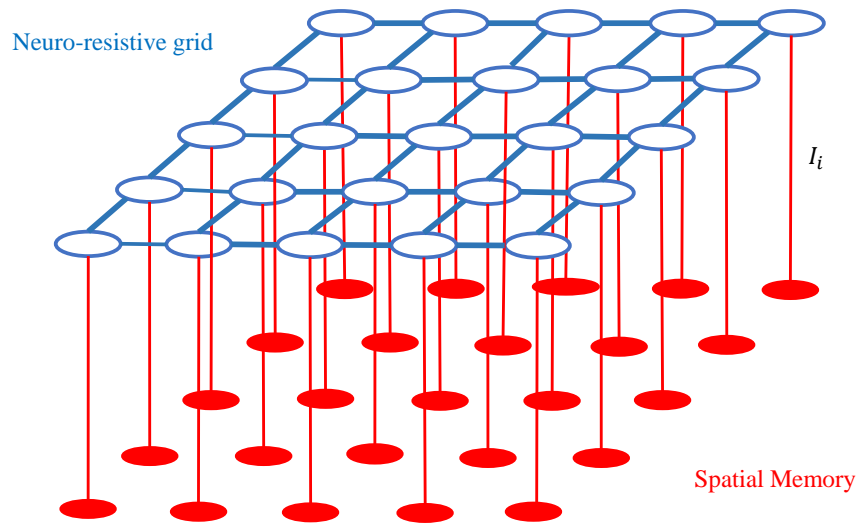


Figure 5.13: Neural implementation of a 5*5 resistive grid and the spatial memory layer (Figure extracted from [Bugmann 1995])

In order to compute the route while ensuring that the final destination is reached in an environment cluttered with obstacles (walls on the map), two steps are performed:

1. A «potential» value of each node i in the resistive grid is calculated and associated with each possible «State» (a «sate» is defined in terms of location in the extracted map at any point, especially with respect to the final destination. In this way, each pixel on the map is a «state», while the final destination is the «target state»). The resulting

potential value y_i decreases as a function of the distance between the corresponding state and the target state which is calculated as in equation 5.1:

$$Y_i = Tf(\sum_{j=1}^m W_{ij} * Y_j + I_i) \quad (5.1)$$

Where, W_{ij} is the weight given to the input from neuron j to i ; Y_i is the output of neuron j ; I_i is an external input from the spatial memory layer used to constrain the value of Y_i , and Tf is a linear saturating transfer function of the neuron i defined as in equation 5.2:

$$Tf(x) = \begin{cases} 1 & \text{if } x > 1 \\ 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases} \quad (5.2)$$

Hence, the node corresponding to the target state is set to a positive potential by adding the external input $I_i=1$ allowing it to act as a current source in the grid. The saturation for negative inputs allows the neurons potential (corresponding to obstacles) to be set to 0 by using a negative input $I_i=-1$. Finally, for all nodes which are not targets, current positions or obstacles, an input $I_i=0$ is defined.

All neurons in the network are updated several times before an equilibrium distribution of the potentials is achieved i.e. the potential values of the resistive grid at time (t) are similar to those of the resistive grid at time ($t-1$). However, when using high resolution grids such as those of a 640x480 pixels image acquired by the robot, the memory size needed for an instantaneous calculation surpasses that of the robot, resulting in an increased computation time. Therefore, in this work, a faster solution has been adopted. It consists of considering that the equilibrium distribution of the potentials has been achieved when the potential of the current state has any value other than 0. The resulting number of iteration is considered to be the minimum number of iteration necessary for reaching a goal state.

2. Once the potential values have been calculated in the resistive grid, finding the path becomes an easy task. By continuously searching among the neighboring states the node with the highest potential, it is possible to perform the transition action to that better state regardless of what the current state is, then searching again for the next better state and so on until eventually the target state is found. There must exist, however, an uninterrupted sequence of permitted states joining the target and the current state.

Sign Sequence extraction Finally, the sequencing of the navigation signs extracted earlier is possible by calculating the position of the recognized signs with respect to the newly generated path.

The operation starts from the starting point and ends in the final destination. To achieve this, certain regular intervals of distance, «checkpoints» (in terms of x - y coordinates), are first demarcated along the path. Beginning at the starting point of the extracted path, for each checkpoint, the distance to all visible signs is calculated. The sign located at the shortest

distance for each checkpoint is stored in a special array. This is done for all checkpoints until the final destination. As a result, the array comprises all the navigation signs visible along and closest to the path at each checkpoint. Then, it is possible to finally extract the right sequence of signs that the robot will encounter on its way. Each sign is stored by following the order given in the memory vector.

Moreover, in addition to their relative $x-y$ coordinates, each «checkpoint» stores the slope value with respect to that of the next checkpoint in order to compute the direction leading from the current checkpoint to the next one. Then, each sign in the memory vector is associated to the direction value given by its corresponding checkpoint.

As a result, the sequence of signs together with their directional meaning is constantly used, in the behavioral module once the robot commences exploration. Therefore, there is no need of recalculating the path or reusing the map once the navigation activity has started.

5.2.3 Behavioral module - Neural structure

Rhizome 2 is composed of two behavioral modules that integrate the information coming from the floor plan analysis in the deliberative module as illustrated in figure 5.14. While the first Behavioral module integrates the sequence of signs, the second behavioral module integrates the direction each sign denotes (left or right). The second behavioral module is connected to the learned output direction group of the SRMA layer in the first behavioral module and sends it directly the associated movement to be performed. Moreover, at the same time that the movements is being performed, a reinforcement signal is activated and learn in the WTA group of the SRMA layer the said association.

Since the details of the first behavioral module have been described in the Rhizome 1 architecture in section 4.2.3 chapter 4, the following section describes the single layer composing the second behavioral module: the Map Direction Information Reflex Behavior (MDIRB) layer.

5.2.3.1 Direction map information (DMI) layer

This layer is in charge of processing the information referring to the sign direction-meaning coming from the deliberative module after floor plan analysis (see figure 5.15).

Hence, when the robot is just in front of a recognized sign, it is able to directly perform the associated movement, which is sent by the DMIRB layer. Additionally, as the robot continues its way towards the next expected sign, it learns the said association in the neural structure.

It is composed of two neural groups as illustrated in figure 5.16 and explained below:

Signs Direction group: This group receives the information concerning the directional meaning of each sign. It is given by two numbers -1, 1 corresponding to the left and right turn respectively. Each neuron is linked to the corresponding left and turn neurons in the *learned output direction neural group* of the SRMA layer, thereby, whenever any of them is activated, it activates the corresponding neuron and when the robot is close to the sign, it performs the corresponding movement. In the meantime, the *reinforcement signal* is activated and the association of the sign with the movement to be performed is learned in the *WTA group* of

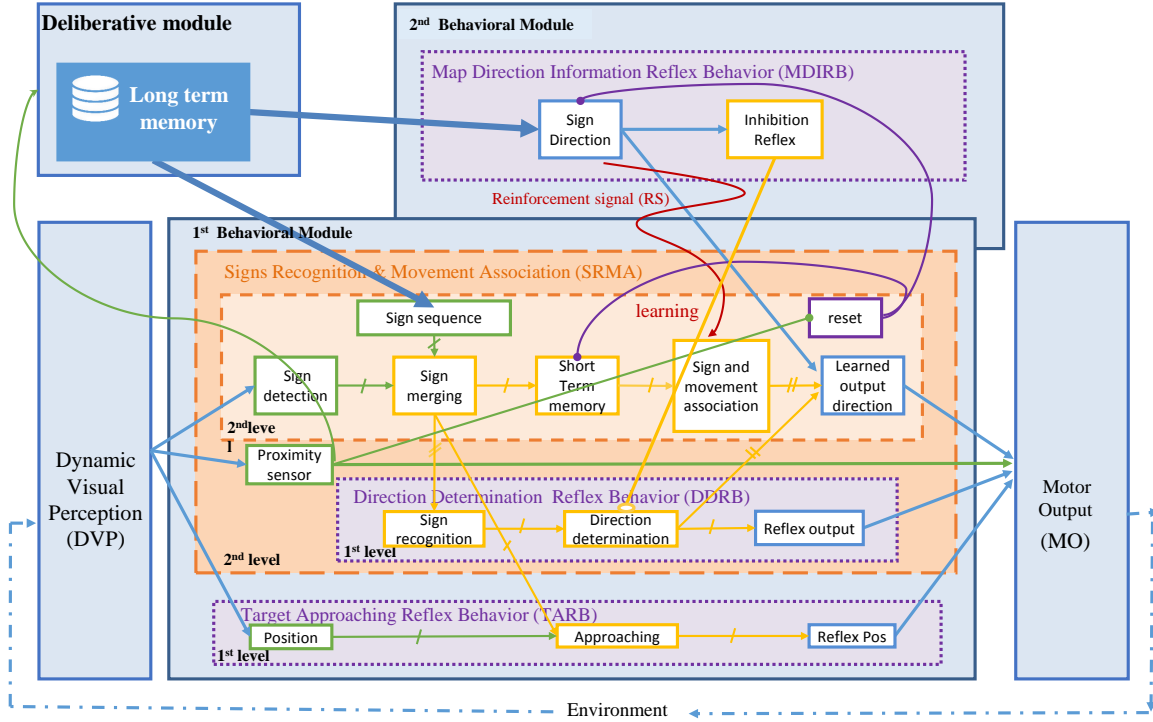


Figure 5.14: General view of the layers in the 1st and 2nd behavioral modules connected to the behavioral module.

the SRMA layer.

Inhibition Reflex group: Since the information referring to the directional meaning is directly extracted from the map, the robot does not need to perform rotatory exploratory movements in order to look for the next expected sign and associate the performed movement as it was explained in Rhizome 1 (see section 4.3). Therefore, whenever any of the two neurons in the *sign direction group* is activated, this single-neuron group is activated and sends an inhibitory signal to the *direction determination reflex behavior layer* of the first behavioral module so that the reflex movements are prevented to be performed.

Finally, the reset neural group of the SRMA layer, which is connected to the *Sign Direction group*, resets the values of the neurons once the corresponding movement has been performed. Thus, the direction of the next expected sign can be assigned.

5.3 Experiments in real environment

The following tests were carried out within the same environmental constraints and conditions of the first experiments presented in Rhizome 1 (see 4). This, with the aim of having the possibility of, not only, evaluating the functioning of the proposed approach but also of being able to distinguish the functionality given by Rhizome 2 with respect to Rhizome 1.

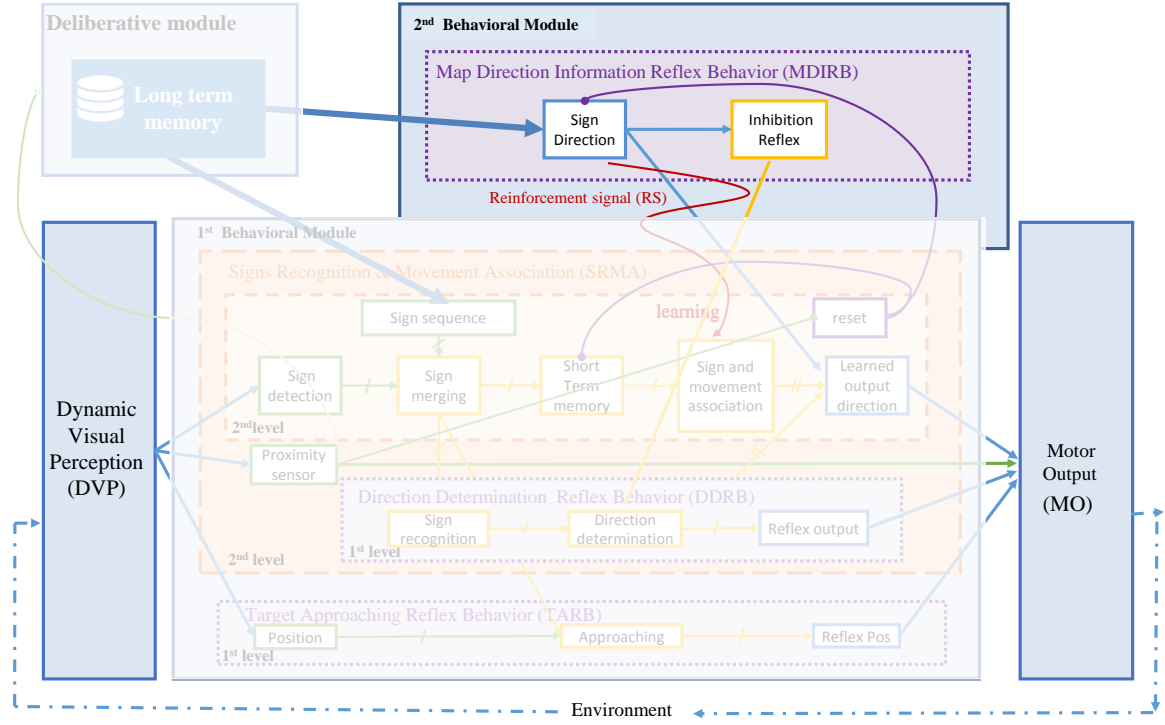


Figure 5.15: General view of the Map Direction Information Reflex Behavior(MDIRB) layer.

5.3.1 Procedure

In order to prove the robustness of the floor plan analysis approach proposed and the efficacy of the navigation task by using a map, a series of test were executed. The robot was expected to compute the path leading to the final destination, extract and memorize a sequence of signs together with the directional meaning associated to each sign and then use them in order to achieve its final destination (figure 5.17).

The map used in this work is a modest floor plan in the simplest way. Only the walls (external and internal) and the navigation signs are here considered. The representation of doors is done by blank gaps.

Consequently, the floor plan of the test environment was first shown to the robot before the navigation process took place so that it could capture some images of the floor plan with its camera and then process the acquired information. Thereafter, the robot was placed at the entrance of the test environment from which it could distinctly see the first expected sign, **A**.

Different image acquisition configurations of the same map were presented to the robot as illustrated in figure 5.18 and according to its capacity at extracting the information after having processed it, the navigation action was expected.

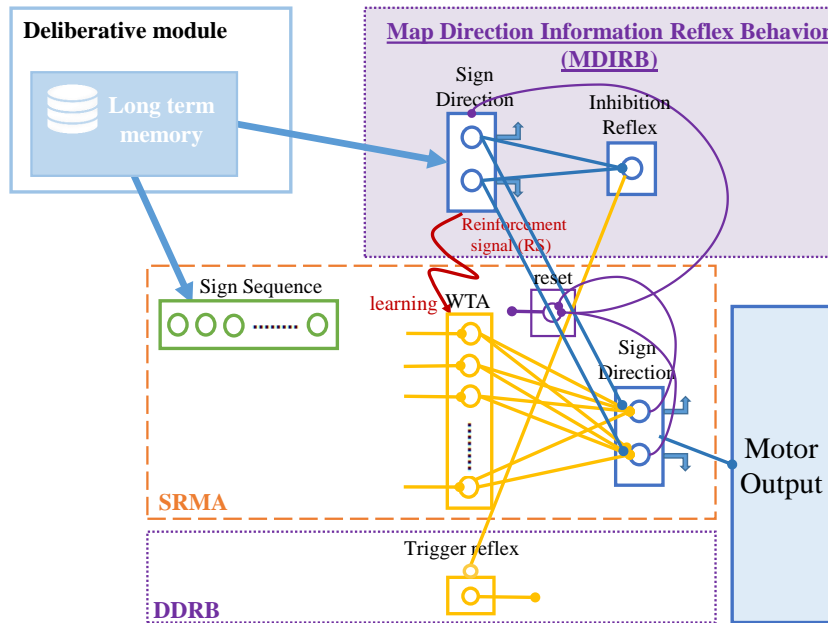


Figure 5.16: Detailed view of the Map Direction Information Reflex Behavior (MDIRB) layer.

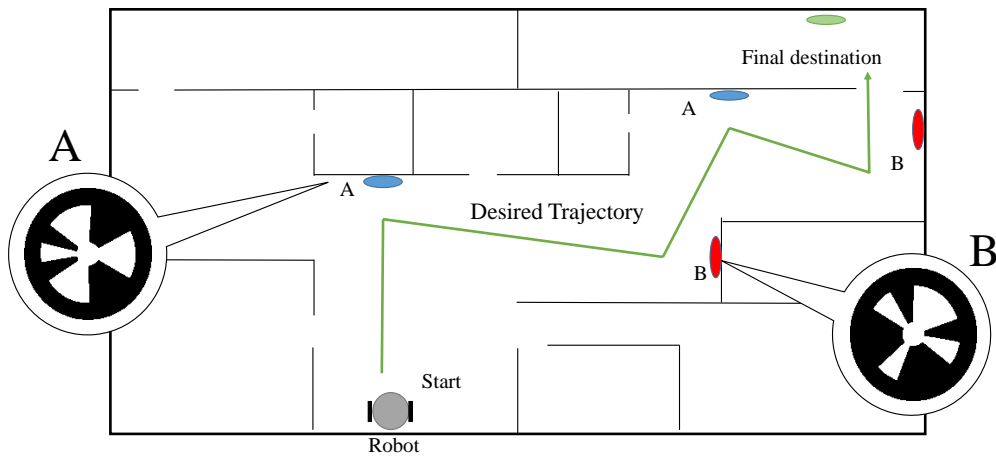


Figure 5.17: Floor plan of the environment. The path trajectory (green line) leading to the final destination is computed after computing a floor plan analysis process.

5.3.2 Results

Deliberative Module

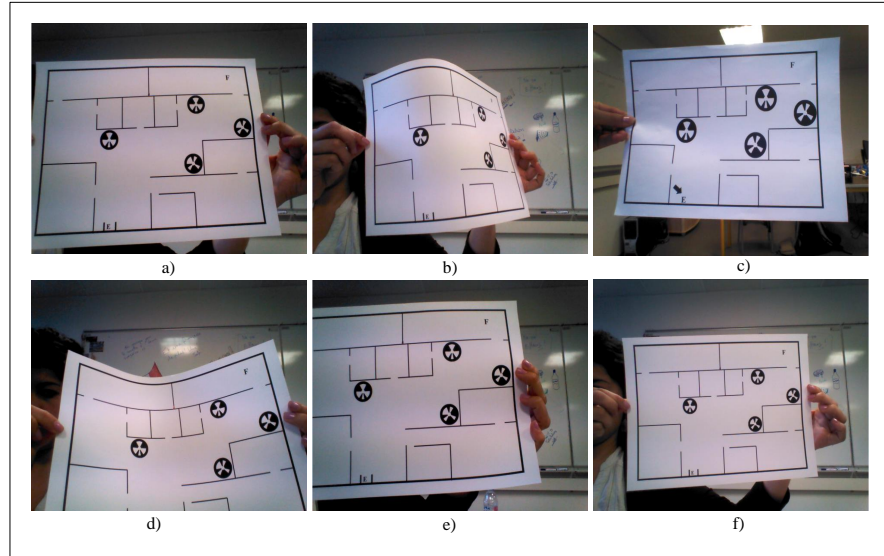


Figure 5.18: Set of raw images of the floor plan corresponding to different acquisition configurations of the same map.

By following the different stages of floor plan analysis explained in section 5.2.2, the robot was able to «read» the map, generate an optimal plan to reach the goal, extract and memorize the sequence of signs (arranged from the closest point to the furthest with respect to the starting point), and the directional meaning each sign denoted.

To begin with, all images of the floor plan taken by the robot were pre-processed in order to remove any insignificant element that may cause problems in further process. To this end, a black top-hat morphological operator with a 7×7 square mask was first applied on the gray-scale image followed by a global thresholding process as illustrated in figure 5.19.

From the figure above, it is possible to see that four out of the six pictures of the floor plan taken by the robot presented some problems. For instance, the floor plan in images d) and e) were outside the camera frame. Image c) instead, seem to be good at a first glance, however, if one looks closer, a gap is in the outside contour of the floor plan. If we go back to the original picture c) in figure 5.18, we can see that the user's finger is the cause of this gap as it is covering the missing part.

Finally, image b) is almost folded which in further process, would prevent the identification of some of the signs. The application of some filters based on the size of the connected regions, permits rejection these imperfect pictures. Therefore only images a) and f) were accepted on which the bounding box was computed (see green square around the floor plan frame of the top images of figure 5.20). Then, the x - y coordinates of the four corners of both, bounding box and floor plan frames were calculated (colorful circles on the corners of both in the same figure) and used in order to compute the perspective correction as illustrated in the bottom part of figure 5.20.

Thereafter, sign detection is performed and the regions around the signs are subtracted from

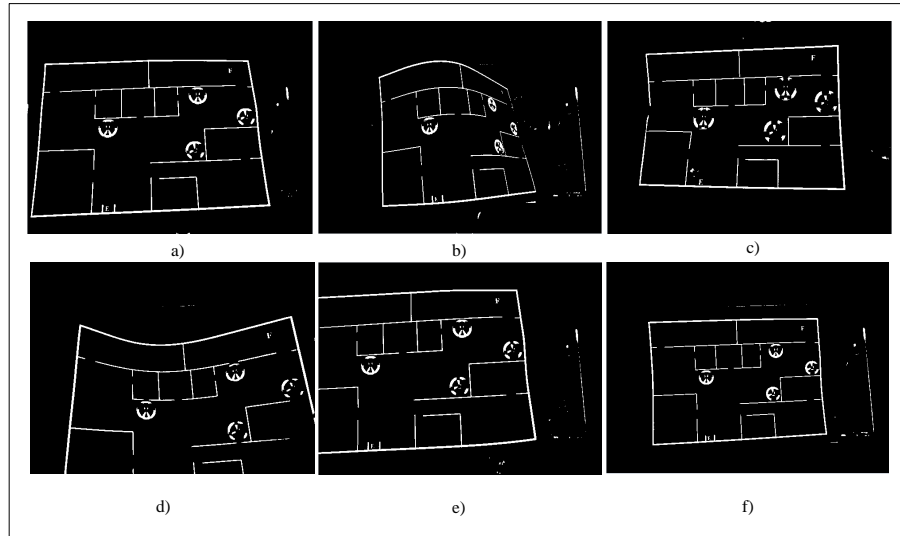


Figure 5.19: Global Thresholding applied to different types of images. Before that, a black top-hat morphological operator with a 7×7 square mask is first applied on the gray-scale image. It removes noise and small object that are not relevant to the analysis in order to improve the accuracy and robustness of the image outcome.

the images. As a result, only the walls remain in the image (see figure 5.21).

The extracted signs are then compared to the Nao Marks signs dataset in order to find their identifier ID letter allowing to distinguish each sign from the others in the environment. From the figure 5.22 it is possible to see that the extracted signs refer to A and B signs which are repeated in the environment.

Finally, the image composed of only walls is used to compute the path that could lead the robot from its starting point to its final one and the sequence of sign is extracted. Figure 5.23 to the left illustrates the checkpoints with the corresponding directions computed on the resulting path plan (colored diamond-shape).

The sequence of signs is thus computed by finding the closest sign to each one of the checkpoints (see middle and right images in figure 5.23).

A summary of the overall processing chain is illustrated in figure 5.24.

As a result, when the map was not placed correctly within the frame of the camera, the robot was able to warn the user to move it. Otherwise, when it was correctly positioned, the robot was able to compute the path leading towards the final destination and thus extract the sign sequence based on the computation of the path and the information of the signs: **{A(turn right), B (turn left), A(turn right), B (turn left)}**.

The average computation time to extract the information was of around 5s.

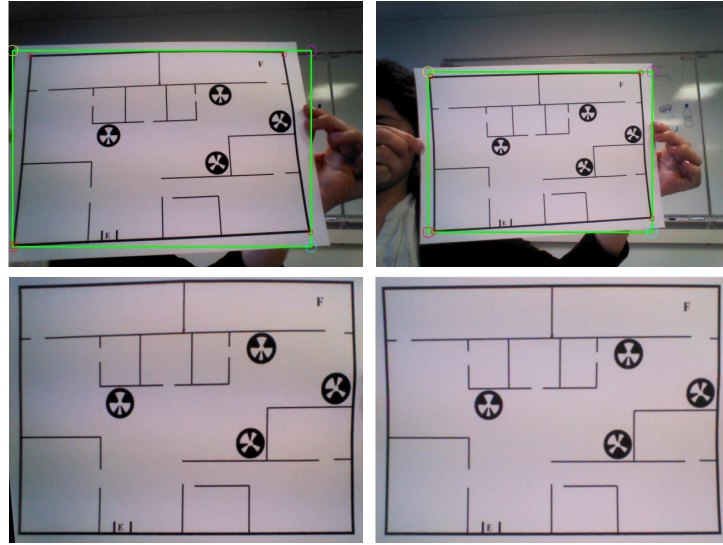


Figure 5.20: Perspective correction computation. A bounding box was computed (see green square around the floor plan frame of the top images). Then, the x - y coordinates of the four corners of both, bounding box and floor plan frames were calculated (colorful circles on the corners of both) and used in order to compute the perspective correction(bottom images).

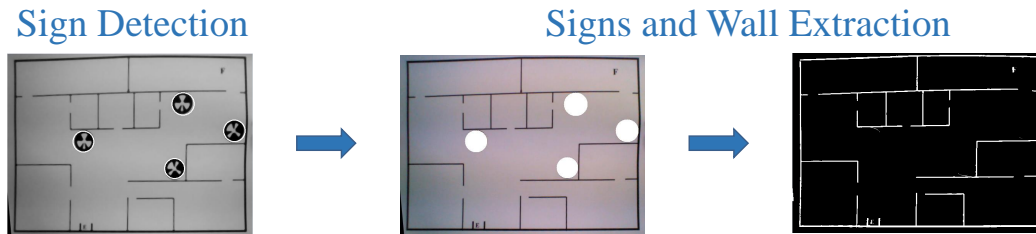


Figure 5.21: Sign detection followed by signs and wall extraction.

Integration into the Behavioral Modules

While navigating the environment (figure 5.25), the robot was successfully able to perform the following intended actions:

- When a sign was detected, it was able to compare it to the corresponding sign from the extracted sequence
- When the comparison gave a negative result i.e. the detected sign did not match the expected sign, the robot ignored the detected sign and continued reflex movements to

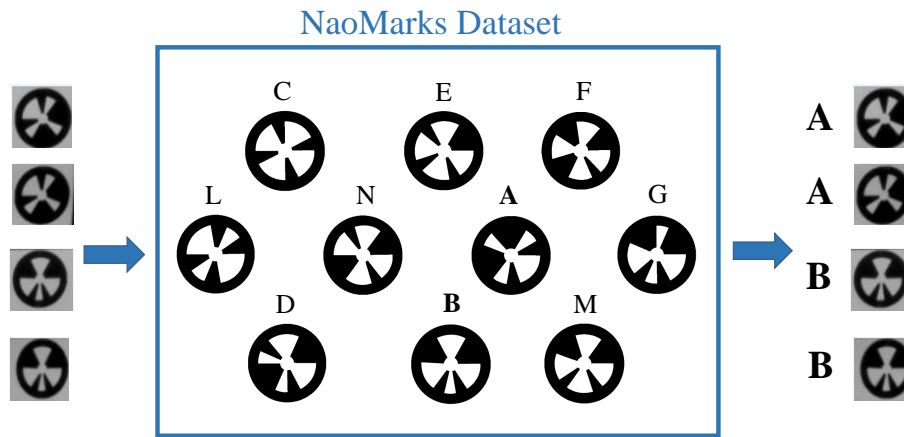


Figure 5.22: Sign classification. The extracted signs are compared to the ensemble of nao mark signs of the dataset in order to identify them by their unique identifier ID letter.

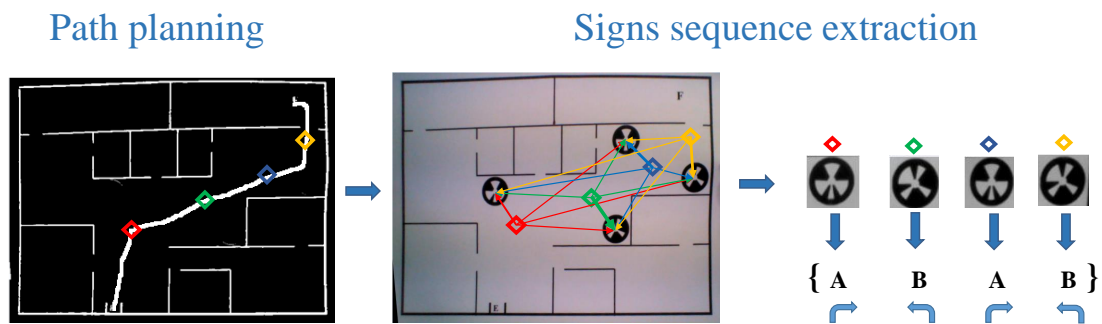


Figure 5.23: left: Path trajectory leading to the final destination with the checkpoints. Middle and right: Sequence of signs computed by finding the closest sign to each one of the checkpoints.

locate the correct sign

- When the comparison resulted in a positive i.e. the detected sign was indeed the expected sign, the robot was able to performed directly the associated movement given by the map and simultaneously learn the association.
- When the expected sign was not in its visual field, it was able to perform a reflex behavior to search for it
- When a sign was faraway, it was able to get closer
- When a sign appeared again, it was able to either recall the learning or directly perform

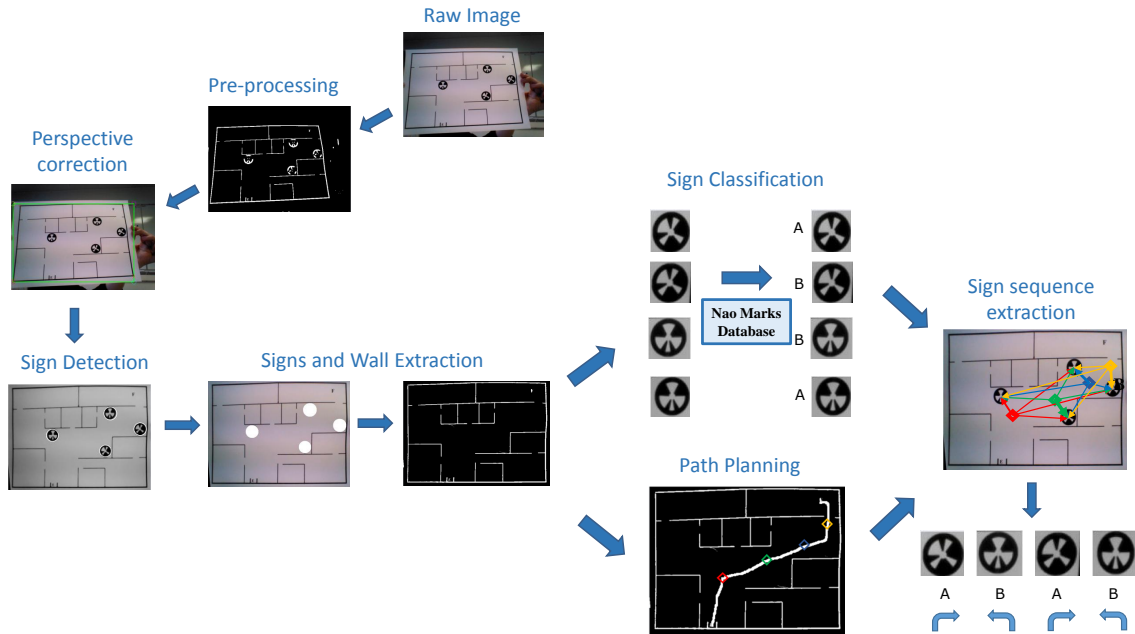


Figure 5.24: Summary of the overall process chain

the associated movement given by the map.

Figure 5.25 illustrates the robot navigating the environment. While the green line refers to the path computed during floor plan analysis, the dotted purple line refers to the actual navigation path executed by the robot in real time.

Figure 5.26 shows a summary of results obtained in the form of the activation of the output neural groups corresponding to the six possible movements that can be performed by the robot (so far), as well as the activation of the reinforcement signal allowing the association learning, over time. The movements were a result of either the recognition, the proximity or the absence of any signs from the extracted sign sequence. The activities are explained chronologically and refer to the descriptions and figures of section 5.2.3.1 and section 4.2.3.2 of chapter 4. In each of the (a, t) plots shown, (a) is the binary activation of each neural group or the reinforcement signal and (t) the time seconds in terms of a PerAc cycle. For the sake of simplicity, the plot labeled *Cumulative target approaching movements combines* all movements undertaken in one go by the robot to approach a particular sign using TARB. For more details of this layer, refer to the result section of chapter 4.

$t_0 - t_8$ When the robot recognized **sign A** at time **t_0** , its corresponding neuron in the SRMA layer of the first behavioral module got activated and remained like this while it was still in the robot's field of view (*Dynamic visual perception* plot in figure 5.26). In the meanwhile, it triggered the activation of the neurons in the TARB layer allowing the robot to approach the sign(see *Cumulative target approaching movements* plot in figure 5.26) until time **t_8** . When the robot was close enough to the sign the proximity sensor got activated at time **t_8** .

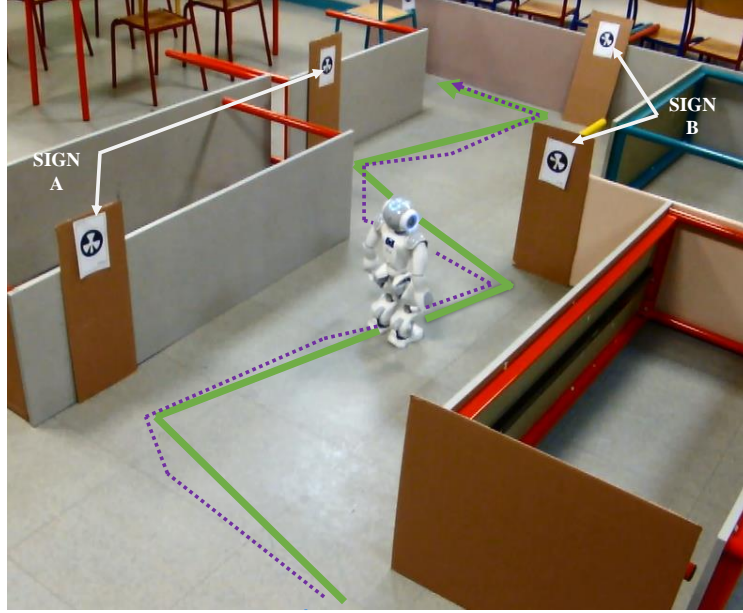


Figure 5.25: Robot navigation. While the green line refers to the path computed during floor plan analysis, the dotted purple line refers to the actual navigation path executed by the robot in real time.

$t_8 - t_9$ Since the associated movement to perform was already known by the robot: **right direction** movement associated to **sign A**, the robot turns directly to the right until detecting **sign B** at time t_9 (see *Right turn* plot in figure 5.26). Simultaneously, the reinforcement signal got activated allowing the robot to learn the said association in the SRMA layer of the first behavioral module (*Learning Association* plot in figure 5.26)

$t_9 - t_{17}$ After having recognized **sign B**, the robot performs the corresponding movements in the TARB layer allowing it to approach the said sign (see *Cumulative target approaching movements* plot in figure 5.26). The activity of the proximity sensor got a positive value at time t_{17} , once the robot was close enough to the sign.

$t_{17} - t_{18}$ Alike sign **A** earlier, current **sign B**'s associated movements was already known by the robot: **left direction** movement associated to **sign B**. Therefore, the robot turned directly to the left until finding the next sign in the sequence : **A** at time t_{18} .

$t_{18} - t_{24}$ After having recognized **sign A**, the robot performs the corresponding movements in the TARB layer allowing it to approach the said sign (see *Cumulative target approaching movements* plot in figure 5.26). When the robot was close enough to the sign, the activity of the proximity sensor got a positive value at time t_{24} . Simultaneously, the reinforcement signal was activated allowing the robot to learn the said association in the SRMA layer of the first behavioral module (see *Learning Association* plot in figure 5.26)

$t_{24} - t_{32}$ From time t_{24} to time t_{32} the same movements explained above are performed. The robot approaches the signs performed by the TARB layer (see *Cumulative target ap-*

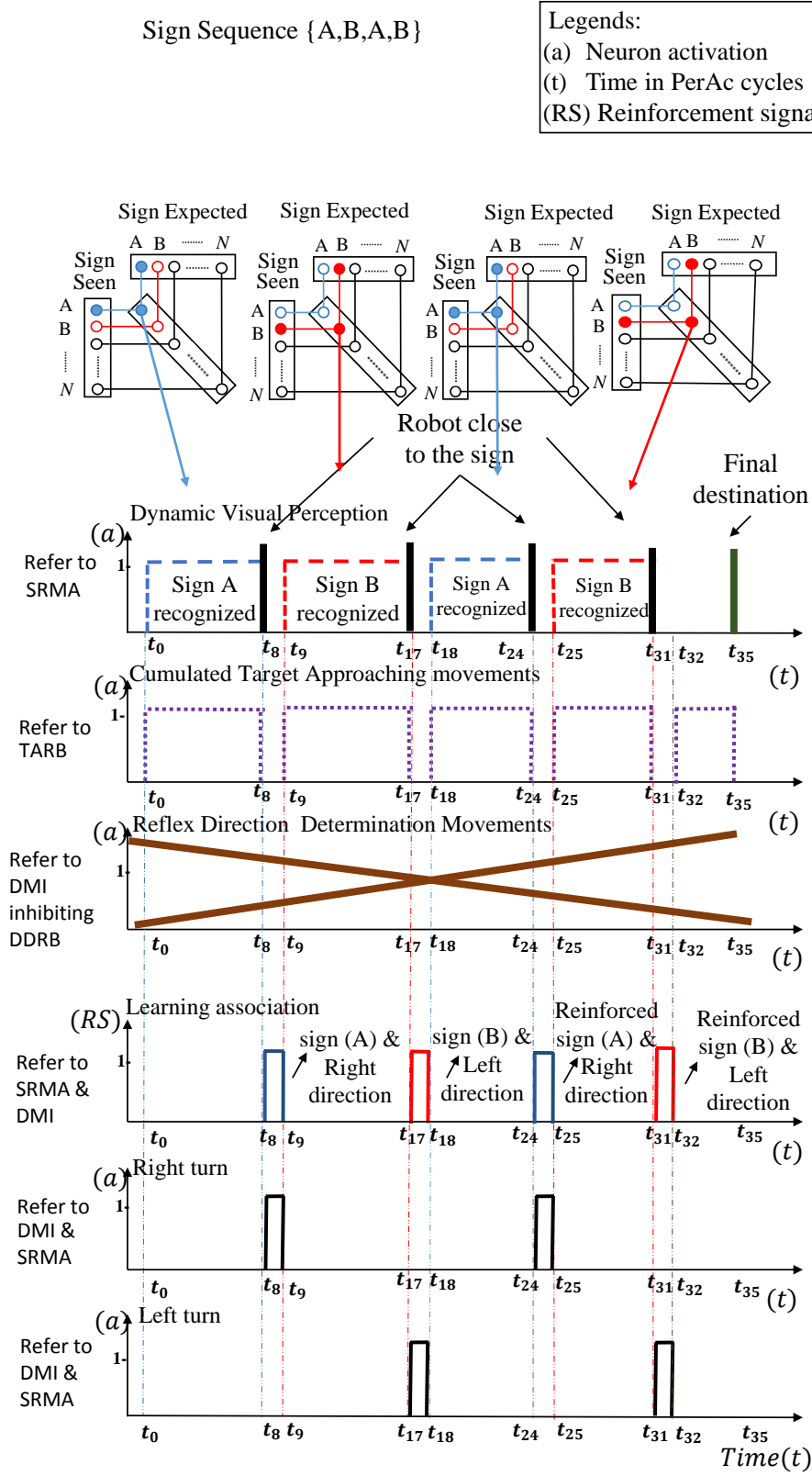


Figure 5.26: Summary of Rhizome 2 results obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal(RS) allowing the association learning, over time. In each of the (a, t) plots shown, (a) is the binary activation of each neural group and (t) the time seconds in terms of a PerAc cycle.

proaching movements plot in figure 4.33) and performs directly the right turn (respectively left) when close to **sign A** (respectively **sign B**) (*Right turn* and *Left turn* plots).

However, as both, **sign A and right turn movement** and **sign B and Left turn movement** associations had already been learned by the SRMA layer, their links are further reinforced by the reinforcement signal (RS).

$t_{32} - t_{35}$ Finally, the robot got to its final destination at time t_{35} after having performed some target approaching movement directed by the TARB layer (see *Cumulative target approaching movements* plot in figure 5.26).

As a result, the robot was able to successfully use the information extracted from the floor plan in order to get to the final destination.

5.3.3 Discussions

Floor plans of buildings can be a very complex depending on the working context. They usually contain details of building elements such as windows, elevations, doors, furniture etc. and text labels referring to information such as the name of the room or the area.

However, in order to prove the concept of robot navigation by integrating a top-down information as explained above such complexities were not essential for this work, at least, at this stage. Therefore, they were excluded in the experimental phase. However, it is assumed that the architecture can be adapted at a later stage to be able to analyze more complex floor plans.

When navigating within the environment, several behaviors emerged allowing the robot to successfully reach its goal. From the results, it can be seen that most of the behaviors were similar to those performed by the robot in the Rhizome 1 architecture (see chapter 4). However, since Rhizome 2 gives the autonomy to the robot of reading by itself a map prior navigation, it was possible to extract additional information (the directional meaning each sign denotes). Therefore, when the robot was closed enough to the signs, it did not need to perform any regular exploratory movements in order to determine the action to take as in Rhizome 1, but instead it directly turn to the corresponding direction given by the computation of the map.

This allowed the robot to arrived at its final destination with some time in advance compared to the navigation performed in Rhizome 1 : t_{35} in figure 4.33 and t_{40} in figure 5.26.

It should be noted though, that even if Rhizome 2 controlled directly the robot towards the left or right direction thanks to the use of the map, the association was still learned in the neuron connections. Similarly, even though the reflex exploratory movements were not performed, the layer can still be activated at any moment where the map is missing.

5.4 Conclusion

In this chapter, a robot uses a chain of image processing techniques for the analysis of a floor plan in real time. Relevant information is extracted (a sequence signs with their directional meaning) and later compared with the dynamic visual information perceived during real

world navigation. Such approach at the same time top-down and bottom-up is merged into a neural system allowing the robot to reach its goal faster in a deterministic scenario.

The Rhizome 2 architecture is composed of a deliberative module and two behavioral modules. Whereas the information concerning the navigation sign sequence is integrated in the first behavioral module, the information corresponding to the «directional meaning» each sign denotes (turn right or turn left) is used in the second behavioral module. The connection between these two behavioral modules enables learning of the association between the sign and its corresponding directional meaning.

Experimental results have shown that the use of a floor plan prior navigation can facilitate and hasten the task of reaching a specific destination inside a building for a mobile robot. Hence, with the correct sequence of signs already captured by the robot, it is ready to begin navigating the building and searching for them in the environment.

Documents and their analyses sometimes offer an indispensable source of information that can be utilized in the robotics field to enhance functionality or substitute other inaccurate/inefficient sources. In return, the dynamic physical abilities and real-time processing can contribute to updating documents, which can be thought as future work. Different behaviors emerging from the interaction between the robot and the environment prove the advantages of a neural approach emulating human behavior in a deterministic scenario.

An eventual scenario could be one where the expected signs are no longer available or visible inside the building (non-deterministic or stochastic scenario). Such situation would force the robot to adapt to the unforeseen changes by reacting accordingly in order to get to its final destination.

Therefore, the next chapter presents Rhizome 3, which was built in order to overcome such situations. To this end, a new behavioral module is built, and added to the overall architecture. It consist in allowing the robot to learn natural landmarks in its surroundings in order to learn places as navigation reference points.

RHIZOME 3: Learning and self-adapting according to unforeseen environment changes

Contents

6.1	General description	159
6.2	Implementation- Rhizome 3 Architecture	163
6.2.1	Overall description	163
6.2.2	Behavioral module- Neural structure	165
6.3	Unitary Experiments in real environment	182
6.3.1	Procedure	182
6.3.2	Set-up	183
6.3.3	Test and Results	185
6.3.4	Discussions	205
6.4	Functional Experiments in real environment	205
6.4.1	Procedure	205
6.4.2	Results	206
6.4.3	Discussions	213
6.5	Conclusion	214

6.1 General description

When working in dynamic scenarios, unforeseen changes are prone to happen. For instance, the *a priori* information (represented in this work by a sequence of navigation signs leading the robot to the final destination) might not be available or visible as it was expected to be found in the environment during real-time navigation. Therefore, the robot needs to be able to cope with such changes while still achieving to its final destination.

Rhizome 3 has been conceived so that the robot can overcome any problem related to such unforeseen changes by learning places as new reference points. Moreover, it enables the robot to perform natural navigation sign recognition when comparing the place to others previously learned (if any) as it navigates. Thus, if the place is not recognized the robot

can learn it as a new one. The architecture allows the robot to learn in an incremental way different places throughout the exploration of the environment whenever it is required.

In this work, a place is characterized by a landmark constellation resulting from a set of patterns referred as landmarks and their corresponding positions perceived by the robot within a panoramic view (360° around itself). Hence, the robot keeps in memory the resulting landmark constellation and learns it as a new place. Each landmark is considered unique within each panoramic view and its position is calculated with respect to a north as represented in figure 6.1.

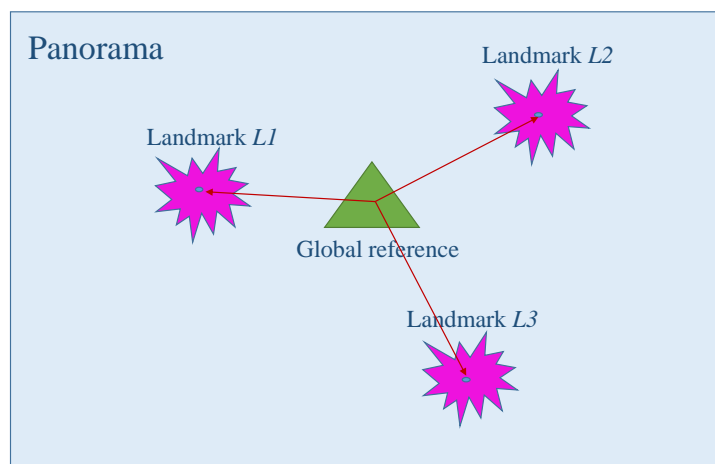


Figure 6.1: Representation of three landmarks considered as unique within a given panoramic view. Their position is calculated with respect to a global reference.

Recognizing a landmark consists then in comparing the current landmarks perceived in the new panorama with those previously learned. If the majority or all of them happened to be similar enough and located at the same or nearby positions, then the place is likely to be the same and thereby it is recognized as such. Otherwise, it is learned as a new one. The decision of knowing if a certain amount of recognized landmarks is enough to state that a given place has been recognized, is given by a vigilance term similar to the one described in the ART model.

Before entering into the details of the implementation of Rhizome 3, let us first have an insight of the overall functional behavior of the navigation process performed by the robot as illustrated in 6.2 and which, is repeated as many times as it is necessary to achieve to the final destination.

It combines the functional behaviors described in Rhizome 1 and that of Rhizome 2 where the directional meaning of the signs can either be known in advance or not depending on the use or not of a floor plan as detailed in chapters 4 and 5. Moreover, the behavior related to the learning and recognition of places when the navigation signs are not visibly available is added to it (see 6.2).

In order to alleviate the representation of functional behaviors, those related to the DDRB

layer from the first behavioral module are not here represented (when there is no use of a floor plan). However, it is clear that it is also part of the overall architecture and functioning. For a better understanding the reader can refer to section 4.2.1 in chapter 4.

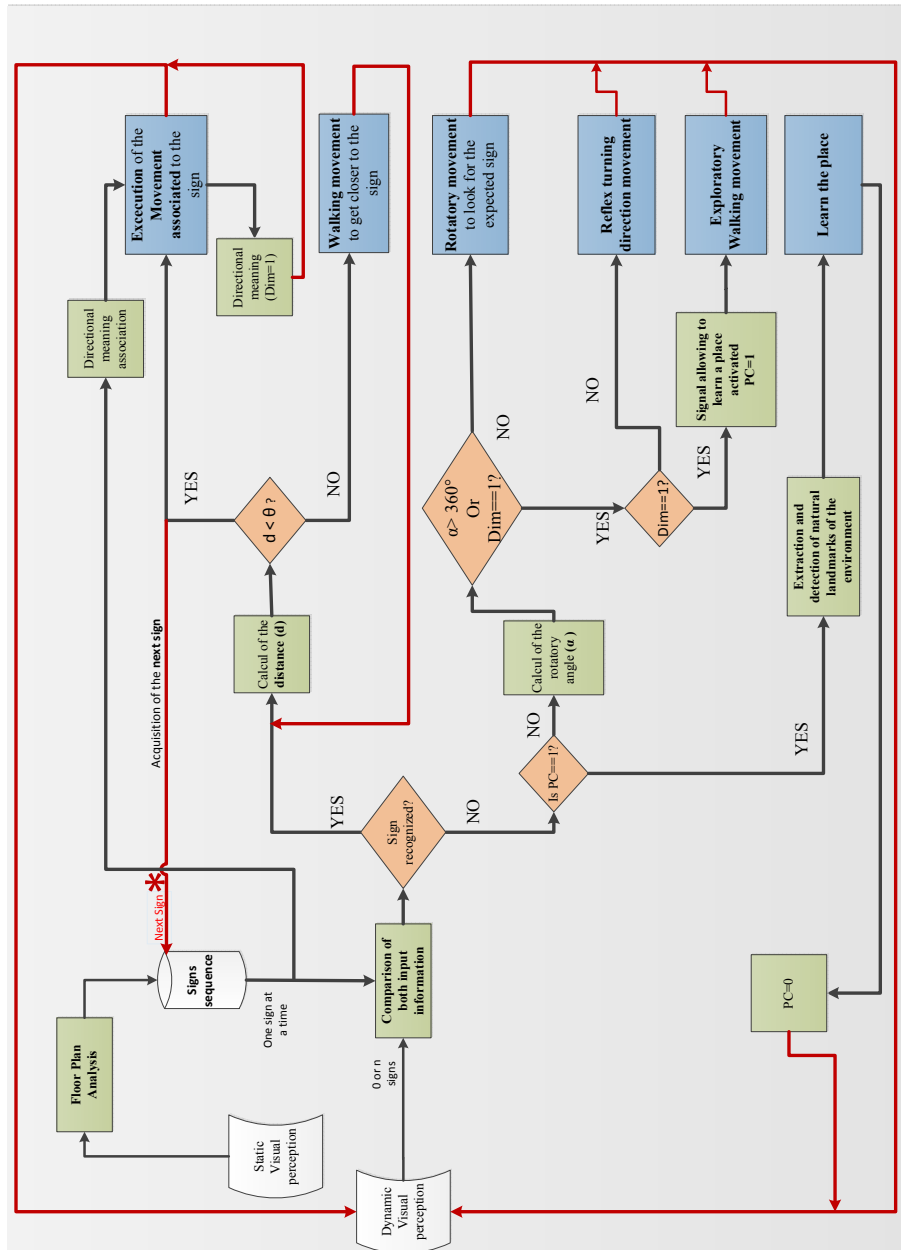


Figure 6.2: Functional diagram of the navigation process behavior allowed by Rhizome 3.

The over all behavior illustrated in figure 6.2 can be explained by following the algorithm:

Input: Two sources of information are used as input in the architecture:

- The dynamic visual perception information, which constantly feeds the system in real-time while the robot navigates the environment. The robot can either detect the artificial navigation signs or extract natural landmarks.
- The static visual perception information extracted from which a sign sequence is extracted. Each sign is given at a time according to what it is expected to be found in the environment. When one of the detected signs matches the current expected sign, the sign is considered recognized and the sign sequence is scanned to obtain the next sign.

Output: a single motor action at a time executed by the robot based on the input information

initialization;

dim=0;

while *the robot has not arrived to its final destination* **do**

 Compare the expected sign to what it sees in real-time;

if *sign is recognized* **then**

 Refer to Rhizome 1 and Rhizome 2 for the detailed description;

 The directional meaning signal «Dim» is activated (Dim=1) after execution of the associated sign given by the map;

else

if *Place Cell signal is activated* ($PC=1$) **then**

 Extract and detect landmark in the environment;

 Learn the place;

 PC=0;

else

 Calculate angle of robot rotation (α);

if ($\alpha > 360^\circ$) *or* ($dim=1$) **then**

if $dim=1$ **then**

 PC=1;

 Perform exploratory reflex walking movements;

else

 Perform reflex turning direction movement;

end

else

 Perform rotatory reflex movements to look for the next expected sign;

end

end

end

end

Algorithm 3: Functional behavior of the Rhizome 3 architecture

6.2 Implementation- Rhizome 3 Architecture

6.2.1 Overall description

The overall architecture integrates the *a priori* information into two organized neural structures that are themselves connected to a third neural structure. Hence, it is composed of four modules as illustrated in figure 6.3. A deliberative module containing the *a priori* information provided to the robot and three behavioral modules: two of them integrating and using the said *a priori* information and the third one dealing with unforeseen environment changes.

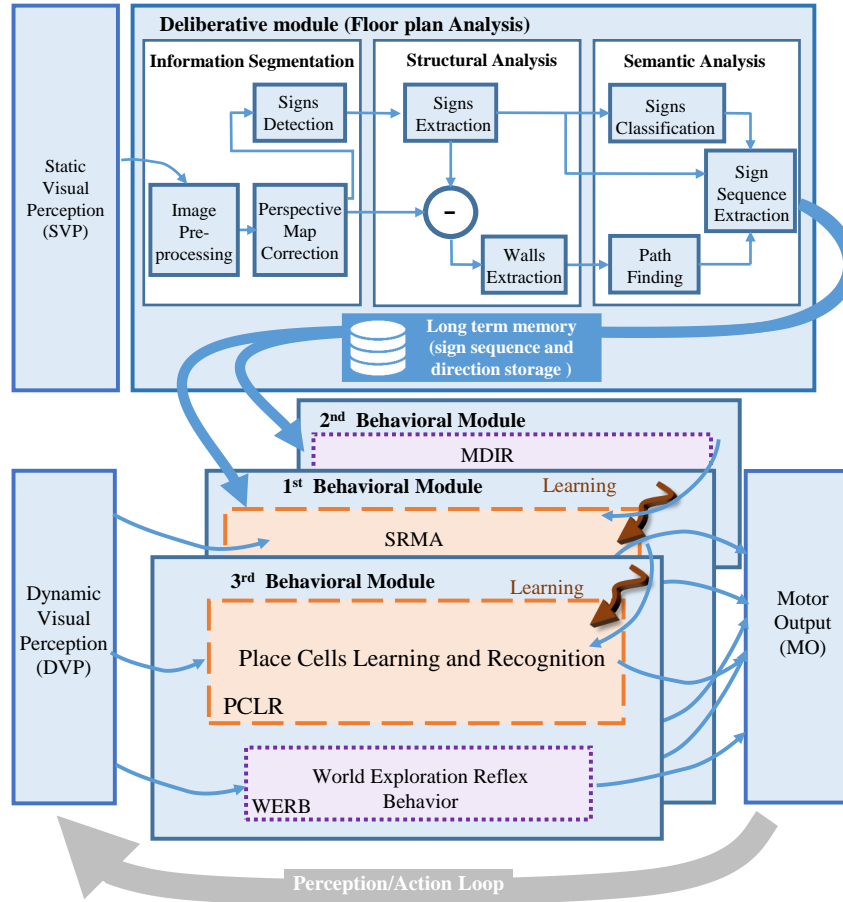


Figure 6.3: Overall view of the Rhizome 3 architecture composed of four modules: one deliberative and three behavioral.

- On one hand, the deliberative module represented by the top box of figure 6.3 stores a succinct information about the environment coding the complete navigation path. It is either provided by means of an external source such as a program command as explained in Rhizome 1 (refer to section 4.2.2 in Chapter 4) or by the analysis of the

environment floor plan as explained in Rhizome 2 (refer to section 5.2.2 in Chapter 5).

- On the other hand, the first two behavioral modules use the said *a priori* information and compare it to the dynamic visual perception. They allow the recognition of a particular sign and learning of the association of its corresponding directional meaning, either by performing rotational reflex movements as explained in Rhizome 1 (refer to section 4.2.3 in Chapter 4) or by directly using the information resulting from the floor plan analysis as explained in Rhizome 2 (refer to section 5.2.3 in Chapter 5).

The third behavioral module instead deals with the unforeseen environment changes presented when the *a priori* information is not available at all or when it does not match to what it is expected to be found during navigation. This module is composed of two layers as illustrated in figure 6.4. It allows the robot explore the environment and learn places as new reference points based on the same perception-action mechanism that the other two behavioral modules follow.

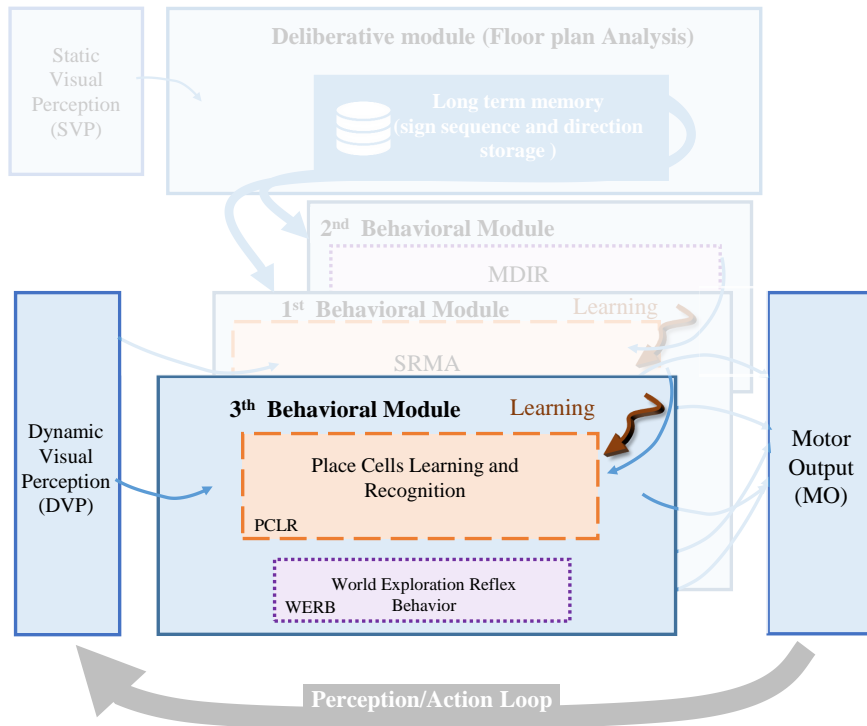
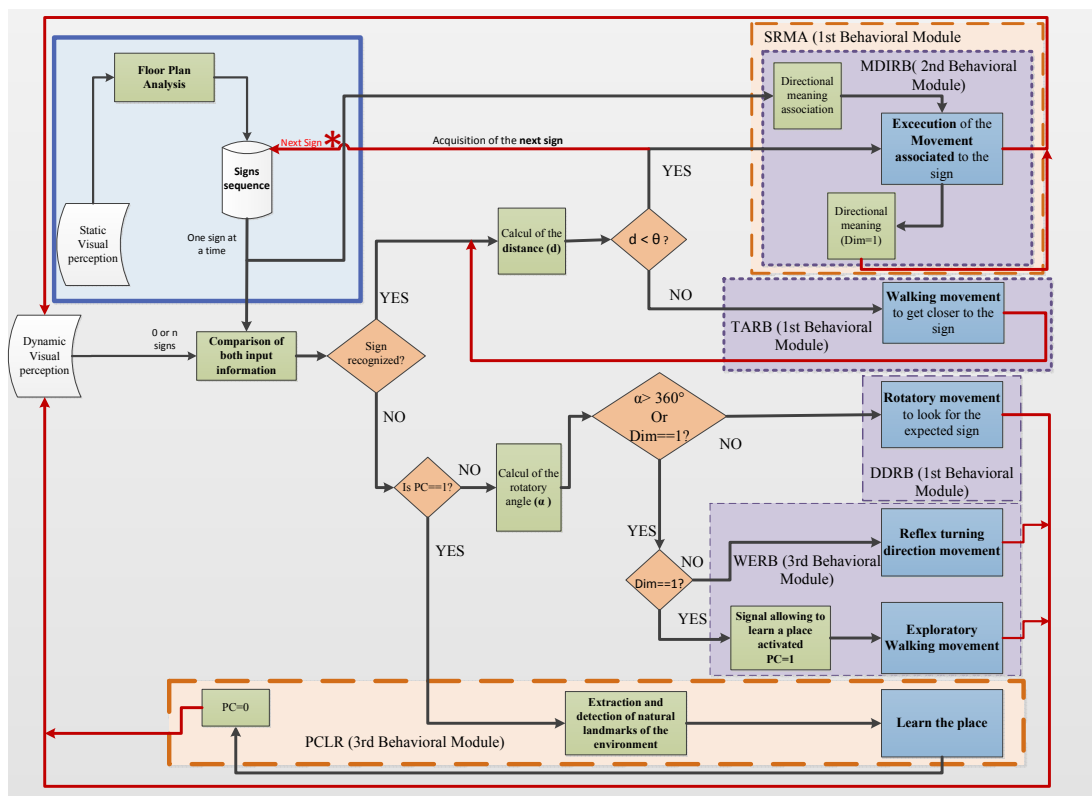


Figure 6.4: Overall view of the third behavioral module composed of a Place Cell Learning and Recognition (PCLR) layer and World Exploration Reflex Behavior (WERB) layer.

The place cell Learning and Recognition layer (PCLR) constantly receives the information coming from the real-time visual perception of the environment as the robot navigates. However, it is only activated when the expected information is not perceived in the environment.

Consequently, if we take the general diagram presented in the general description, each module intervenes on each of the following actions as illustrated in figure 6.5.



Overall description

A robust visual place recognition algorithm needs to combine descriptive, discriminative and generalization properties. Therefore, in order to capture all these properties, Gaussier [Gaussier 2000] has proposed a biologically inspired approach based on the representation of place cell for recognizing places within the navigation environment. Such model allows recognizing a place resulting from a particular merging configuration of two flows of information: *what* and *where* [19]. While, the former refers to the identification and recognition of patterns perceived in the visual scene of the place, the latter refers to the spatial location of the same.

Hence, it is necessary to, on one hand, describe the perceived patterns in a distinctive way and on the other hand, find their respective location within the scene.

The PCLR layer allows learning and recognizing different places within the environment by reproducing the same place cell model. However, it addresses the recognition problem in a manner that differs from the model proposed by Gaussier mainly in two points:

- in the procedure used for detecting and extracting the landmarks and;
- in the internal computation of the neural components of the *what group* and its consequences on the *PrPh merging matrix group*.

Figure 6.6 illustrates the overall state-of-the-art place cell model (blue frame) and the contributions made in our model (orange frame).

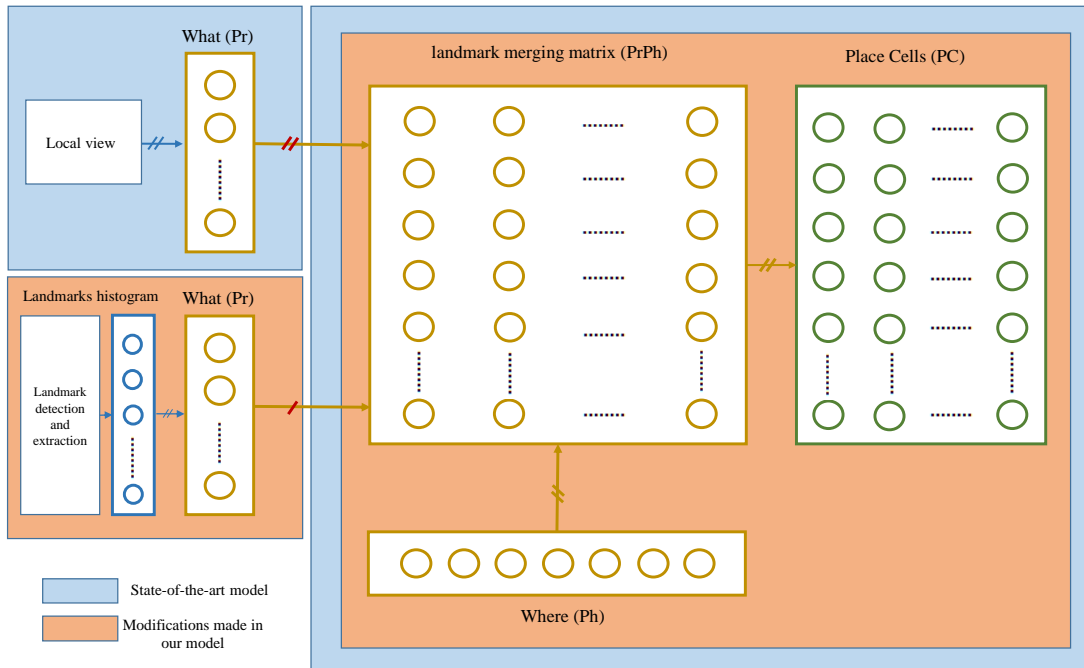


Figure 6.6: Overall state-of-the-art place cell model (blue frame) and the contributions made in our model (orange frame).

A summary of both models with respect to the two aforementioned points is given below. For a better understanding of the Gaussian model and the corresponding equations, the reader can refer to the state-of-the-art section.

1. *Local view or landmark extraction process:*

- **Gaussian model**

The gradient of the image is used and convolved with a difference of Gaussians (DoG) filter in order to detect robust focal points such as corners or edges at a particular spatial (low) resolution. Then, in order to reduce the computation time, a simplified process averages and weights all images columns for the points near the center of a column and the resulting one-dimensional signal is differentiated. Thus, local maxima is used as the focal point.

Finally, a 32*32-pixel area around each of the focal points is extracted and considered as a local view (landmark), which undertakes a log-polar transformation so that they can be invariant to small rotations and scale variation.

- **Our model**

All images undertake two-classification process. First, a visual bag of words model together with the SIFT local descriptor [Lowe 2004] is built and used in order to describe distinctly the salient features of all images. Second, the same features are clustered according to their proximity in terms of their local position. The resulting groups are considered as landmarks and each of them is compared to the others by computing the norm of the difference between the clusters center of the feature describing them. A supplementary neural group is added in the model, in order to store the information, represented as a histogram, coming from the created landmarks.

2. *Neural computation:*

- **Gaussian model**

This model supposes that a landmark cannot be found twice in the same panoramic view as it would not succeed in knowing which azimuth is associated to each landmark.

Therefore, the *what group* recruits a different neuron for each perceived landmark. If for instance, a same local view is perceived from two different angles when learning different places, two different neurons are recruited. Then, in order to allow multiples interpretations of the same local view, an activation function f^{RT} is used. This competition mechanism enhances the built-in generalization capability. All neurons in the *what group* and the *where group* are connected to all neurons in the *PrPh matrix*. Such connectivity allows learning in the connection weights linking both, the *what* neuron and its associated neuron in the *PrPh matrix* and the *where* neuron and its associated neuron in the *PrPh matrix*.

Since several neurons may be activated in the *what group* because of the multiple interpretation mechanism, the activity of the neurons in the *PrPh matrix* is

computed by searching for the maximal value among the values resulting from the product between the activated *what* neurons and their corresponding connection weight.

- **Our model**

Contrary to the state-of-the-art model that allows the multiplicity of recruited neurons for each perceived landmark regardless their angle of perception, our model tackles this problem by allowing the system to learn incrementally the perceived landmarks.

This is possible by modifying the recruitment of neurons in the *what* group.

- Every single landmark perceived from the first place of the exploration environment encode a different *what* neuron.
- From the second place to the end of the exploration, each perceived landmark is compared to the ones previously learned. If after comparison, the recognition value is superior to a *vigilance term* :
 - * the corresponding neuron gets activated with the resulting recognition value;
 - * otherwise, a new *what* neuron is recruited and learned.

As a result, there is no need to code two or more neurons for the same landmark even if it is seen from another angle, allowing to reduce to a fair quantity the number of neurons necessary to be recruited in the *what* group.

Moreover, while all neurons in the *where* group are connected to all neurons in the *PrPh* matrix as in the state-of-the-art model, the connection between the *what* neurons and the *PrPh* matrix neurons are done in a one-to-one fashion. More precisely, a single neuron in the *what* group is connected to its corresponding row of neurons in the *PrPh* matrix with an initial weight value of one as illustrated in the figure 6.7. By consequence, learning is only performed on the weights between the *where* neurons and their associated neurons in the *PrPh matrix*.

The implementation of how a place cell is constructed and learned is described in the next section by detailing each of the components of the proposed model.

As illustrated in figure 6.8 , each place is characterized by a set of landmarks surrounding the robot within its panoramic view (360°) and the *information extracted from all landmarks* is used as the input of the network system. Each natural landmark is associated to a neuron from the *what group (Pr)* which, is set as its unique identifier. The relative position information of each landmark is associated to a neuron from the *where group (Ph)*. Then, the information coming from both groups converges on a two-dimensional array of neurons, which keeps in memory the resulting value of all the landmarks perceived in the same panorama.

As a result, a landmark constellation is formed in the *landmark merging matrix group* leading to the learning of a new place by recruiting a new neuron in the *Place Cell group*.

1. Input information

(a) Natural Landmark detection and extraction

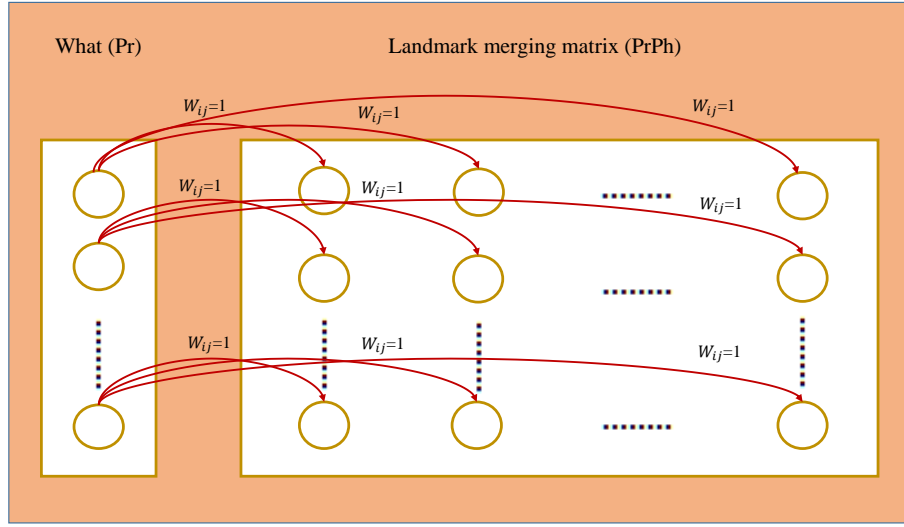


Figure 6.7: Neural connection between the *what group* and the *PrPh matrix*. A single neuron in the *what group* is connected to its corresponding row of neurons in the *PrPh matrix*.

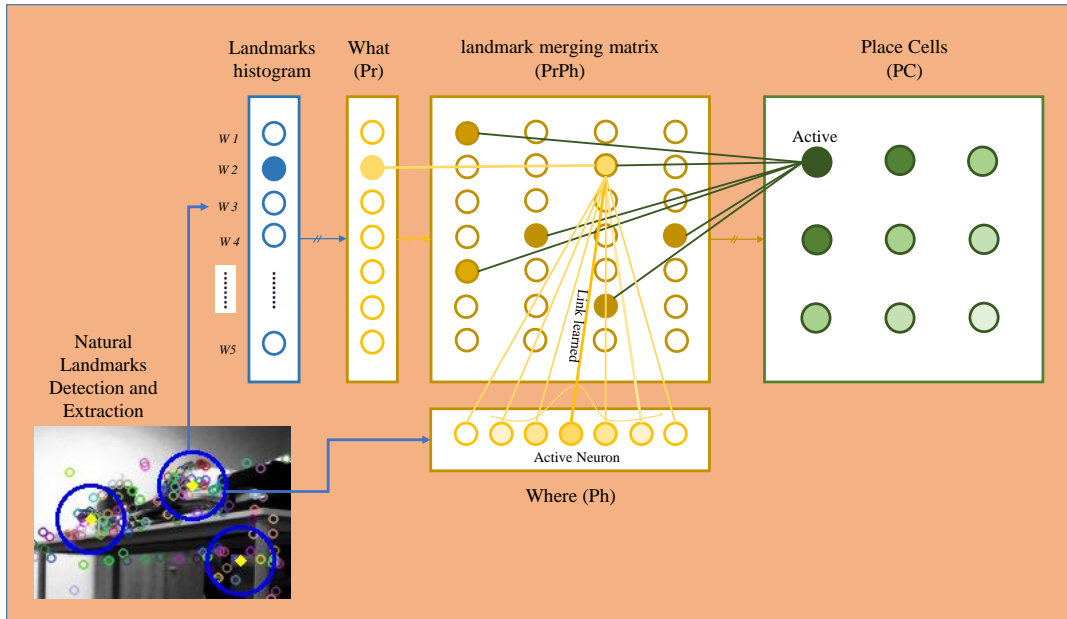


Figure 6.8: Proposed Place cell model. Composed of five neural groups. The information extracted from all natural landmarks in the panorama view are the input of the layer.

Overall description The information extracted from all natural landmarks in the panorama view are the input of the network system. It is divided into two

flows of information in order to feed the *what* and the *where* groups:

- The first flow of information is related to the description of each landmark given by a vector of key-points it comprises of. It is gathered in the *landmark histogram neural group*, which is connected, to the *what group*.
- The second flow of information is related to the position of the same. It is given by the X-coordinate of the landmark within the image with respect to a reference global point and it is immediately fed into the *where group* to be processed.

Hence, in order to obtain such flows of information, it is necessary to first detect and extract the landmarks out of the images perceived by the robot within its panoramic view.

In this work, the natural landmarks are considered as patterns described by their distinguishable features (key-points). Hence, in order to detect these patterns, all images acquired by the sensor camera are first pre-processed and transformed into gray color space.

For each image, the salient features along with their descriptors are computed. Since the viewpoint of a pattern can drastically change from one position to another, the SIFT descriptor is here used. As a result, each feature is described as unique and differ from one another according to the composition of their visual characteristics.

Then, the images are converted from the set of features into a bag-of-words representation in order to match the appearance of the current scene to the trained data. To this end, the process is performed for two different sets of images: The template images to build the vocabulary and the query images to build, detect and recognize the landmarks during the robot run-time navigation. Such process is usually used in the literature before sending the information to the classifier for learning.

However, since our aim is to search natural landmarks within the images, we need to go beyond this point to find the interest regions that can be considered as potential landmarks. Therefore, the algorithm is extended by using another type of cluster (relative to the spatial information distance) over the same descriptors in the same images: landmark clustering.

As a result, each spatial cluster is considered as a natural landmark, represented by a certain number of descriptors spatially close to each other and belonging, each of them, to different clusters from the vocabulary.

Figure 6.9 illustrates the overall process, which is described in detail below.

A. Vocabulary Construction: As previously said, in order to build the vocabulary a set of template images is used in figure 6.10. In fact, they serve as the model from which all the new images will be based on.

The set of template images used here depicts the navigation environment used for the test from different viewpoints from which the robot could perceived the

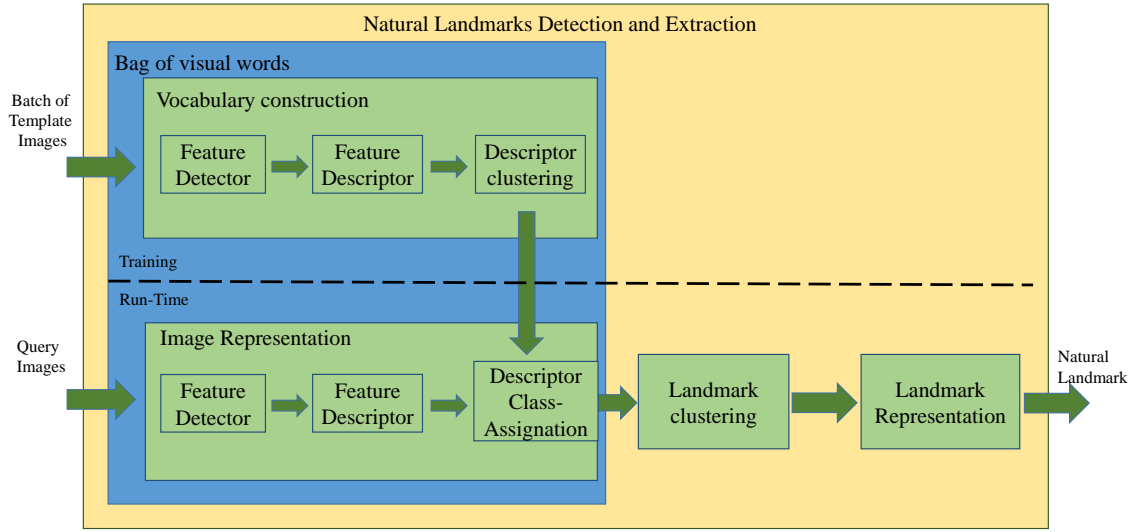


Figure 6.9: Overall description of the Natural landmarks Detection and Extraction process. While a batch of template images is used to build the vocabulary, the query images are used to learn and recognize the natural landmarks.

environment as well as other places with similar objects. The choice of using a wide set of images was made in order to improve the quality of recognition.

For each template image, all interest features (key-points) are detected and their descriptors are computed with the SIFT descriptor. Then the k-means algorithm is used in order to group all the descriptors into k different clusters according to their similarity. This whole process is only performed once; therefore, it was computed before the robot navigation. At the end, the clusters centroids are stored and used as inputs in the following stage.

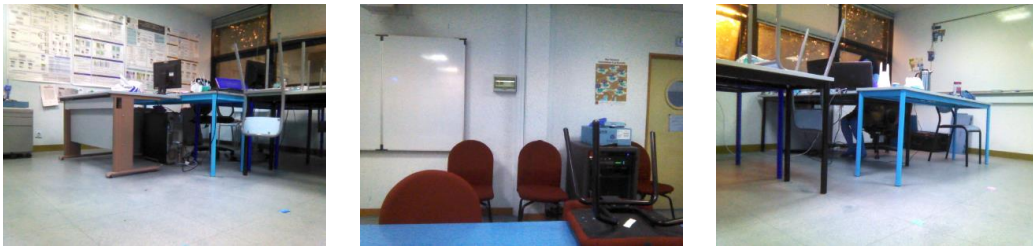


Figure 6.10: Example of Images serving to construct the vocabulary

B. Image Representation: Once the vocabulary has been built, the extracted features from the robot's perceived images of the navigation environment (query

images) are compared and assigned to the clusters centroids of the vocabulary according to their similarity by computing the k-means algorithm. Thus, for each new image the following process is performed.

- First, all key-points are detected and their descriptors are computed with the SIFT descriptor.
- Each key-point is associated with one of the clusters based on their descriptor similarity (see left image of figure 6.11) by computing the Euclidean distance between the 128-element descriptor and the 128-element centroid descriptors of the vocabulary.

From that point, the image can easily be represented by a histogram(right image in figure 6.11), which is computed by counting the number of key-points, assigned to each descriptor class d . The number of classes corresponds to the total number of clusters in the vocabulary and bins in the histogram.

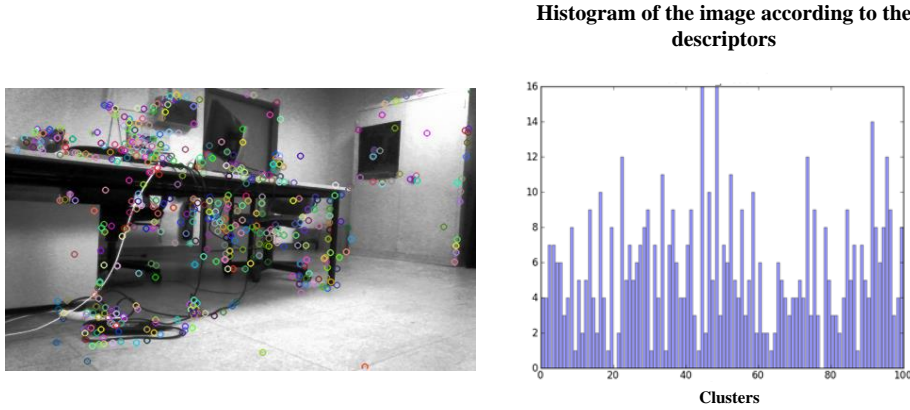


Figure 6.11: Left: a query image with the ensemble of extracted key-points, each one assigned to the cluster corresponding to the nearest centroid of the vocabulary. Right: Image represented by a histogram of a total number of clusters in the vocabulary (here 100).

The integer label to which each extracted key-point belongs according to the descriptor similarity cluster (*descrip_label*) is stored in a vector « Similarity_vector (V_s) » of size equal to the total number of extracted key-point (m). This number can vary from image to image.

$$V_s = \{descrip_label_1, descrip_label_2, descrip_label_3, ..., descrip_label_m\}$$

C. Landmark Clustering: The construction of the set of landmarks is achieved by grouping the extracted key-points from the image into n different clusters, but this time according to their spatial distance (see figure 6.12 to the right). This is done by using the Euclidean distance and the x and y coordinates of each key-point. The n number of clusters corresponds to the total number of landmarks in each image composing the panoramic view. Akin to the *Similarity_vector*, a

vector «distance_vector (V_d)» is created. It stores the integer label to which each extracted key-point belongs according to the spatial clusters ($dist_label$). The vector's size depends on the number of extracted key-points (m), which can vary from image to image.

$$V_d = \{dist_label_1, dist_label_2, dist_label_3, \dots, dist_label_m\}$$

Then, in order to build each natural landmark according to both information so that they can be easily recognized among others, the following algorithm 4 was performed.

It is composed of labeled key-point descriptors belonging to a specific spatial cluster ($dist_label$) based on the proximity to the centroid of the said cluster.

As a result, each spatial cluster considered as a «natural landmark», comprises of a certain number of key-point descriptors close to each other and belonging, each of them, to different clusters of the vocabulary.

The whole image can then be represented by a histogram, but this time, considering the spatial clusters representing the natural landmarks as illustrated in figure 6.12.

Input: $V_d = \{dist_label_1, dist_label_2, dist_label_3, \dots, dist_label_m\}$
 $V_s = \{descrip_label_1, descrip_label_2, descrip_label_3, \dots, descrip_label_m\}$
Output: $AllLandmarks = [L_{dist_label_1}, L_{dist_label_2}, \dots, L_{dist_label_n}]$
 with, $L_{dist_label} = [descrip_label_1, descrip_label_2, \dots, descrip_label_i]$
 initialization;
 $AllLandmarks = []$;
 $i = 0$;
while $i < total\ number\ of\ landmark\ clusters\ (n)$ **do**
 for each descriptor integer label ($descrip_label$) **in the similarity_vector** (V_s) **do**
 $L_{dist_label} = []$;
 for each descriptor integer label ($dist_label$) **in the distance_vector** (V_d) **do**
 if ($dist_label = i$) **then**
 $L_{dist_label} = [store\ descrip_label]$;
 end
 end
 end
 $i = i + 1$;
 $AllLandmarks = [Store\ the\ L_{dist_label_i}\ list]$
end

Algorithm 4: Algorithm allowing to compute the construction of natural landmarks based on their descriptors similarity and spatial distance.

Figure 6.12 to the left, shows that the number of key-points belonging to each cluster (natural landmark) varies from one another. Such disparity makes it diffi-

cult to perform recognition of landmarks, as the comparison among them cannot be done by the same standards.

Therefore, a landmark representation process allowing describing all landmark under the same based needs to be considered.

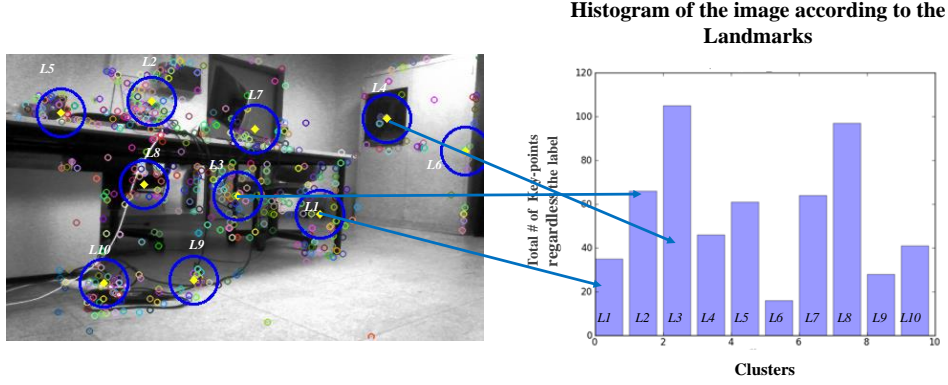


Figure 6.12: Left image: Natural landmarks resulting from spatial clusters composed of a certain number of key-points. The number of key-points varies from one another. Right image: histogram of the image according to the natural landmarks.

D. Landmark Representation: Based on the process of image representation explained in the above section, the construction of a histogram representing the landmark with the same visual bag of words seemed to be the most suitable option for us.

Therefore, each landmark is represented by a histogram composed of as many bins as there are words in the SIFT-based vocabulary (k clusters). Then, for each landmark defined by a spatial cluster, each bin (w) counts the total number of key-points within the landmark that are associated with the cluster corresponding to that bin.

$$L_{dist_label} = [W_1, W_2, W_3, \dots, W_k]$$

As a result, the landmarks have the same number of parameters to be compared to, regardless the number of key-points comprising them. Thus, each landmark is characterized by three elements in the image space as follows:

- the performance of the histogram translated into a k -sized vector;
- the x-coordinate corresponding to the relative position of the landmark-cluster center with respect to the horizontal axis of the image;
- and the y-coordinate corresponding to the relative position of the landmark-cluster center with respect to the vertical axis of the image.

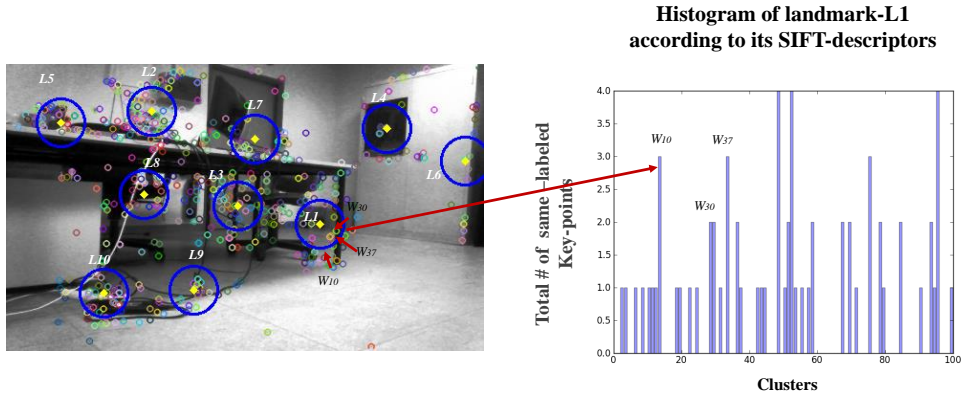


Figure 6.13: Left image: Several natural landmarks resulting from spatial clusters composed of a certain number of key-points. Each key-point has a descriptor label based on the clusters created during the vocabulary construction. Right image: histogram of landmark L1 according to its descriptors.

(b) Landmark histogram group

This group is in charge of storing the landmark histogram extracted above. It comprises of as many neurons as there are clusters in the vocabulary. Each neuron acts as a bin (w) and its activity value corresponds to the number of key-point belonging to the given cluster bin. If for instance, there is no key-point belonging to a given cluster inside the landmark, the activity value of its corresponding neuron is zero. Figure 6.14 illustrates the performance of the landmark histogram expressed in the landmark vector L1 which is injected in the landmark histogram group by assigning the bin values to the corresponding neurons.

Once the activities of all neurons have been assigned, the resulting vector is then injected into the *what group* to be either learned or recognized (see next section *what group*) and the activity values are all restarted to zero so that the same process is repeated for each landmark in the panorama view.

2. Landmark and azimuth learning and recognition

(a) What group

The *what group* is composed of a sufficient number of neurons to encode the total amount of landmarks that can be found in a given exploration environment. Given the lifetime of any robot for learning an infinite number of places in different environments, this number can be considered infinite.¹

Two different procedures have been implemented, one for learning and the other for recognizing. The first one consists in recruiting and learning one neuron for

¹In the case of this work, the *what group* is composed of 200 neurons to learn 9 different places in a test environment. Even though not all of them were used for the exploration of the test environment, it did not have an impact on the learning performance. On the contrary, it gave itself a leeway in case of need to learn new landmarks.

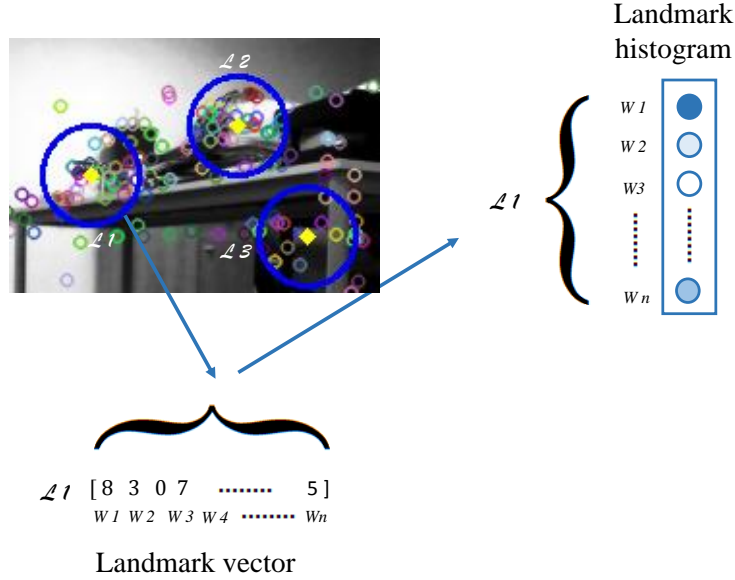


Figure 6.14: Performance of the landmark histogram expressed in the landmark vector $L1$ which is injected in the landmark histogram group by assigning the bin values to the corresponding neurons.

every landmark perceived in the panoramic view of a given place. The recruited neuron is thus associated to the landmark by performing a one-shot learning. The synaptic connection weights W_{ij} between the input neurons and the *what* neurons are first initialized to zero and then modified according to the following learning rule.

$$\Delta W_{ij} = a_i * a_j \quad (6.1)$$

Where, the current neuron activity $a_j = 1$ when the *what* neuron is recruited and $a_j = 0$ otherwise; and a_i the activity of the landmark input neuron (see figure 6.15) which value is defined by the landmark histogram group. Every time, a new neuron is recruited, a global variable adds up the number so that the next neuron from the *what* group can be recruited and so on.

The second procedure consists in comparing the landmarks of a new place to those already learned to find out if the landmark can be recognized. To this end, all the neurons activities are computed accordingly to the following equation:

$$a_j = 1 - \left(\frac{\sum_{i=0}^{L_i} |W_{ij} - a_i|}{L_l} \right) \quad (6.2)$$

Where, the current activity neuron a_j gets the maximum value when the Euclidean distance between the synaptic weights W_{ij} and the input value a_i are nil. L_l is the constructed landmark l .

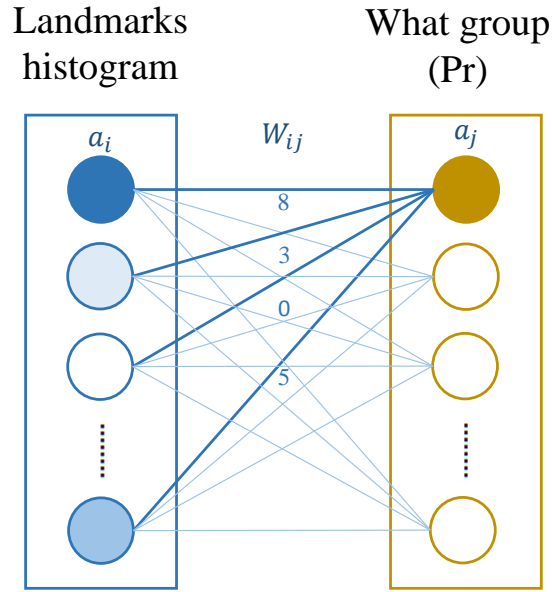


Figure 6.15: Activity of the landmark input neuron which value is defined by the landmark histogram group.

The resulting values determine how similar the current landmark is to the previously learned. In fact, the smaller the Euclidean distance is, the more similar both landmarks are. Therefore, according to the above equation, the neuron with the highest activity value among all the what neurons is likely to be the most similar one.

Then, by verifying if the neuron has already been associated to another landmark and if its activity value is bigger than a «vigilance term»², it is possible to conclude that the landmark has been recognized. If on the contrary, its activity value is rather small (inferior to the vigilance term) or the neuron has not yet been associated to any landmark before, it is likely that the current landmark has not been recognized and by consequence it has to be learned as a new one by following equation(6.1).

(b) **Where group**

The *where group*, is composed of a limited amount of neurons encoding a preferred direction covering in all the total 360° of the panoramic view in order to compute the angular position of each landmark. As the angular position (azimuth) of each landmark is given by its distance in pixels along the x-axis with respect to a reference point, the preferred directions are also given in pixels and along the x-axis. For every landmark position, the *where group* is expressed as a non normalized gaussian activity profile:

²Vigilance term = 0.95 in this work

$$a_j = \exp - \frac{(\alpha_l - \mu_j)^2}{2\sigma^2} \quad (6.3)$$

Where, α represents the azimuth of the L_{th} landmark and μ_j the preferred direction of the neuron j . Each preferred direction is computed as:

$$\mu_j = \frac{dist}{2} + k * dim, k \in \{1, 2, 3...dim\} \quad (6.4)$$

Where, $dist = \frac{x_{total}}{dim}$ is the distance in pixels between each preferred direction, x_{total} the total number of pixels in a row on the panorama view and dim the number of neurons in the *where* group.

The activity of all neurons in the where group is computed with the same α_l value and since their preferred direction differs from one another, only the closest neuron to α_L results with the maximum activity value and consequently gets to encode the landmark position. Moreover, in order to compute the product between the *what* and *where groups* for merging purposes, the final activity value of the neuron encoding the landmark position is set to 1 just as it was done with the activation value of the winner neuron in the *what group*. Thus, the next time the robot sees the same landmark at the same place, the corresponding neurons from both groups will have the maximum activity value.

Now, assuming that the robot has navigated a given distance from the learned place, the chances of recognizing the same landmark in a position far away from the learned one are slim. In fact, as the positions of the landmarks are computed with respect to a reference point, no matter how much the angle of perception of the robot varies, that if the landmark happens to be recognized it will most certainly be perceived within the vicinity of the place where it was learned before. Therefore, the neurons close to the one encoding the landmark position are also considered and associated to a certain extent to the same place.

To this end, equation (6.3) is applied to all neurons, but instead of $(\alpha - \mu_j)$, the equation uses $(i_0 - i_n)$.

Where i_0 represents the index of the neuron encoding the landmark position and i_n the index of its eight surrounded neurons.

Then, it is possible to set an activity value that decays in function of the distance of each neuron to the encoding one. All neurons beyond a threshold distance will be set to zero. Thus, if the landmark is recognized but is seen in another position, the value of recognition resulting from the product of the *what* and the *where* neuron will be slightly smaller than if the landmark was recognized at the exact learned position.

In summary, the neuron encoding the landmark position is the one with the maximum activity value after applying a Gaussian function and its neighbours are set to a value that decays in function of their distance to it as illustrated in figure 6.16.

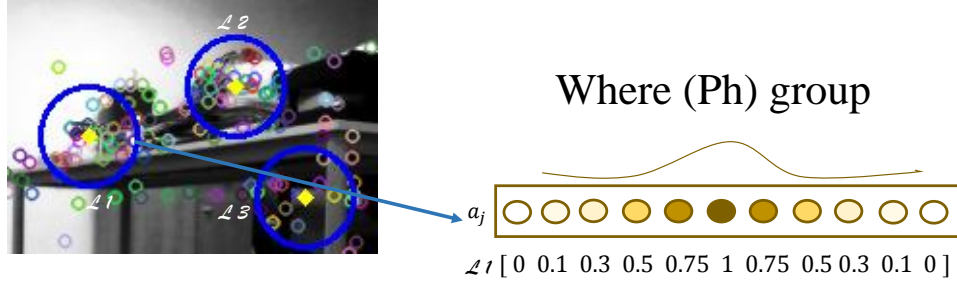


Figure 6.16: Activation of the *where* group. The *where* neuron encoding the landmark position is the one with the maximum activity value after applying a Gaussian function and its neighbours are set to a value that decays in function of their distance to it

Finally, since the analysis is carried out for each landmark in the panorama, two one-dimensional vectors keep in memory the resulting neurons of each landmark description and position respectively. Only one neuron from the *what* group and one from the *where* group with their maximum values encode one single landmark of the panorama view. Hence, at the end of the panorama analysis, both one-dimensional vectors contain the overall information necessary to build the landmarks constellation.

3. Spatio-temporal merging constellation build-up

In order to learn a place, the robot needs to keep in mind the information of all landmarks perceived from its point of view. However, since the analysis of the place, which is given by the analysis of the landmarks within the panorama view, can only be done in a sequential mode (the system cannot recognize several landmarks in parallel), it is necessary to keep in memory the overall information.

Thus, in order to suppress the sequential aspect of the scene exploration, a matrix of neurons stores the information of all landmarks perceived in the panorama view. In fact, the information coming from both *what* and *where* groups of each landmark converges into the *landmark merging matrix* allowing a spatio-temporal merging. As a result, a landmark constellation is formed allowing to learn a new location by encoding a neuron in the *Place Cell* group.

The Place Cell group is depicted by a neuron configuration such that it represents the topology of the environment as illustrated in figure 6.17. In the example, a test environment in a square shape suggests a matrix configuration.

(a) landmark merging matrix (PrPh)

The number of neurons the *PrPh landmark matrix* comprises of corresponds to the number of neurons allowing encoding as many landmarks as possible within the

to know how far a perceived landmark is from the learned position. The further the landmark is from the learned position, the smaller the result value will be.

The weights between the *where* neuron and its associated neuron in the *PrPh landmark matrix* are initialized to 0 and learning is performed as follows:

$$\Delta w_{jk}^{Ph-PrPh} = 1 \quad (6.6)$$

Where, $w_{jk}^{Ph-PrPh}$ is the connection weights between the j_{th} neuron azimuth to the k_{th} *PrPh* neuron.

Each landmark perceived in the panorama is related to a *what* and a *where* neuron which relation is stored in the *PrPh landmark matrix*. Therefore, The *PrPh* neuron activity results from the product of both inputs and it is calculated by the following equation:

$$I_{ij} = (Pr_{ai} * w_{ik}^{Pr-PrPh}) * \max_j(Ph_{aj} * w_{jk}^{Ph-PrPh}) \quad (6.7)$$

Where, Pr_{ai} and Ph_{aj} are the activity values previously computed and correspond to the maximum activity values of both *what* (*Pr*) and *where* (*Ph*) groups respectively. The activity value of the *where group* is stored in a vector and corresponds to the set of activity values of the encoding neuron and its neighbours. Therefore, a max operator allows choosing the maximum value coming out of the product operation of the activities with their corresponding connection weights. $w_{ik}^{pr-prph}$ and $w_{jk}^{ph-prph}$ are the connection weights between the i_{th} neuron landmark and the j_{th} neuron azimuth respectively to the k_{th} *PrPh* neuron.

Thereafter, the time integration process is achieved by repeatedly performing the described process and setting the activity of the corresponding neurons to the resulting product value:

$$a_{ik} = I_{ij} \quad (6.8)$$

Figure 6.17 illustrates the formed landmark constellation as a result of the product between the *what* and the *where group*. Subsequently, the landmark constellation activates with the maximum value a neuron in the *Place Cell group* whereas the others get a decreasing value with respect to the distance.

From the figure, it can be seen that even though the neuron with the most active value in the *where group* has not been learned, the second neighbor *where neuron* to the right, which was learned previously, results activated (see the connection value).

4. Place location learning

- (a) **Place Cell group** The place cell group comprises of a number of neurons encoding different locations in the environment. Each different location or place is characterized by a unique landmark constellation formed by the all landmarks perceived within the panorama view (process described above).

The whole group of neurons is connected to all neurons in the *PrPh landmark matrix* and their initial synaptic weights values are set to zero.

When learning a new place, a neuron from the *PC group* gets to be recruited. Then, for all neurons in the *PrPh landmark matrix*, if the activity of each of them happens to be superior to 1, its synaptic weights connecting the said *PrPh* neuron to the recruited *PC* neuron is set to one. This can be summarized by equation 6.9.

$$\Delta w_{ik,p}^{PrPh-PC} = \begin{pmatrix} 1 & \text{if } a_{ix} > 0 \\ 0 & \text{otherwise} \end{pmatrix} \quad (6.9)$$

Then, the activity of the P_{th} neuron of the *PC* group is calculated as follows:

$$a_p^{pc} = \frac{1}{w_p} \left(\sum_{i,k=0}^{M-1,N-1} a_{ik}^{PrPh} * w_{ik,p}^{PrPh-PC} \right) \quad (6.10)$$

With $w_p = \sum_{i,k=0}^{M-1,N-1} w_{ik,p}^{PrPh-PC}$, the sum of all connection weights that are linked to the place and M, N are the row and column dimensions of the matrix.

The resulting activity reaches the maximum value 1 when the robot is at the exact same location where it has learned the place before, and it decays exponentially with respect to the distance of the robot's current location to the learned one. This *a priori* generalization property allows the system to still activate a place cell when the robot is within the vicinity of the learned location.

6.3 Unitary Experiments in real environment

6.3.1 Procedure

In order to prove the viability and robustness of the place cell model proposed for learning and recognition, a series of experiments have been conducted according to the two types of learning presented in the state-of-the-art: traditional (Batch) learning and incremental learning.

On one hand, batch learning consists in performing learning and recognition in two phases. Firstly, all the concerning test places are learned off-line. Thereafter, the robot is positioned again at the previously learned places and the recognition activity is calculated.

On the other hand, in the incremental learning the robot discovers new places as it navigates the environment. Learning and recognition take place online. This requires, comparing the currently perceived panorama with those already learned (if any) and based on the resulting recognition rate, learning it as a new place.

The main idea of these experiments is to analyze the recognition rate of several places that have been learned (both traditionally and incrementally) by the robot. By verifying the performance when recognizing places, it would be possible to characterize the robustness of the proposed approach regardless the type of learning. Moreover, the differences resulting from the comparison of both types of learning would allow us to highlight their advantages

and drawbacks and thus, let us to favor the use of one over the other according to different scenarios.

However, before executing the above-mentioned tests, some preliminary experiments needed to first be executed in order to find the parameters that suited the most to accomplish a successful recognition.

Throughout the construction of the layer, different decisions had been taken according to our working context. To this end, a variety of parameters essential to the task needed to be considered such as the number of snapshots surrounding the robot, the number of landmarks per images or even the number of clusters attributed to build the vocabulary, etc. While, some of them (judged to not have a big impact on the final result) were fixed empirically or by performing some non-relevant tests, other such as the parameters allowing the SIFT algorithm to detect the key-points in the environment, have undertaken a thorough series of tests and are explained below.

6.3.2 Set-up

In order to validate the proposed approach, the place cell learning and recognition layer was implemented in the humanoid robot NAO and tested in a 2.5m x 2.5m room space.

To this end, a simple scenario was created such that a certain number of positions scattered all over the test environment was chosen to code different places within it.

Figure 6.18 depicts the environment on which different tests were carried out.

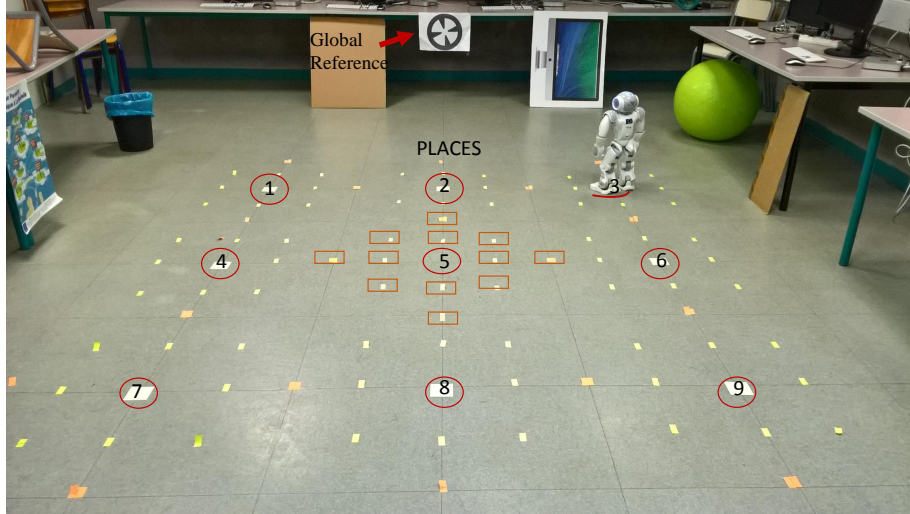


Figure 6.18: Environment on which different tests were carried out. A certain number of positions scattered all over the test environment was chosen to code different places within it.

- There are as many places as it is required to map the overall navigation environment. In this work, we consider nine different positions representing nine main places within

the test room (see red circles in figure 6.18) which are 2 meters away from each other.

- Moreover, additional places among the main ones have also been considered in order to test the learned places. More particularly, those surrounding the middle place (see orange squares around place 5 in figure 6.18). The distance among them is of 50 cm.
- A place is characterized by a set of landmarks perceived by the robot within a 360° panorama.
- The 360° panorama is obtained by taking snapshots of the environment with the camera's robot while rotating around itself. In overall, twelve overlapping images are considered enough to complete one panorama and thus represent a place. Figure 6.18 illustrates three of the twelve snapshots of the environment.



Figure 6.19: Exemple of three images out of the twelve used to create the panorama of a given place.

- Each image is composed of several key-points as illustrated in figure 6.20, which number depends on **the parameters of the SIFT algorithm**. The number varies from image to image. All key-points have been assigned to different descriptors clusters from the vocabulary according to their similarity. In total, 100 clusters have been chosen to build the vocabulary. This number results as the average of the total number of key-points computed in several images.

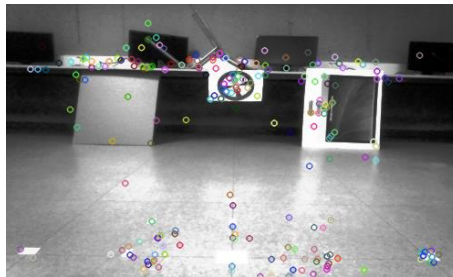


Figure 6.20: Image key-points computed by employing the SIFT algorithm.

- The same key-points are themselves clustered according to their spatial proximity representing the natural landmarks (see pink asterisk-shape in figure 6.21). **Ten spatial clusters** were considered good enough to cover a whole image and fixed for all images regardless the number of key-points detected in the image. This can be optimized at a later stage if required by adjusting the number of landmarks according to the number of key-points detected per image. However, the results being quite convincing, we did not go beyond this point.

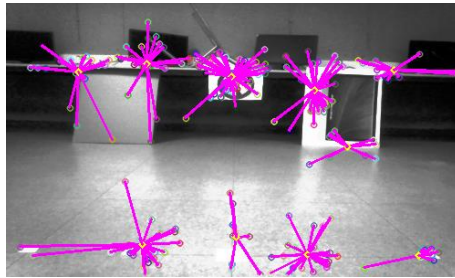


Figure 6.21: Representation of the natural landmarks clustering the key-points according to their spatial proximity.

- However, since each image overlaps with the previous and next images, a filter cutting the overlapping borders was used. Therefore, only the landmarks detected within the **60 % of the center of the image** were considered (see figure 6.22).
- As a result, each landmark is characterized by three elements:
 1. the performance of the histogram translated into a 100-sized vector;
 2. the x-coordinate corresponding to the relative position of the landmark-cluster center with respect to the horizontal axis of the image;
 3. and the y-coordinate corresponding to the relative position of the landmark-cluster center with respect to the vertical axis of the image.
- The position of the landmarks is calculated with respect to a north (here an artificial landmark has been placed as the global reference point). The computation is performed by only considering the **x-coordinates** of both landmarks and reference point as illustrated in the figure 6.23. The **y-coordinates** can be used to improve the system but this is not considered in this model.

6.3.3 Test and Results

When learning and recognizing a place, different configurations can be possible. For instance, a single place can be learned and then its recognition rate is computed by placing the robot at different places surrounding it. Additionally, many places can be learned beforehand, and

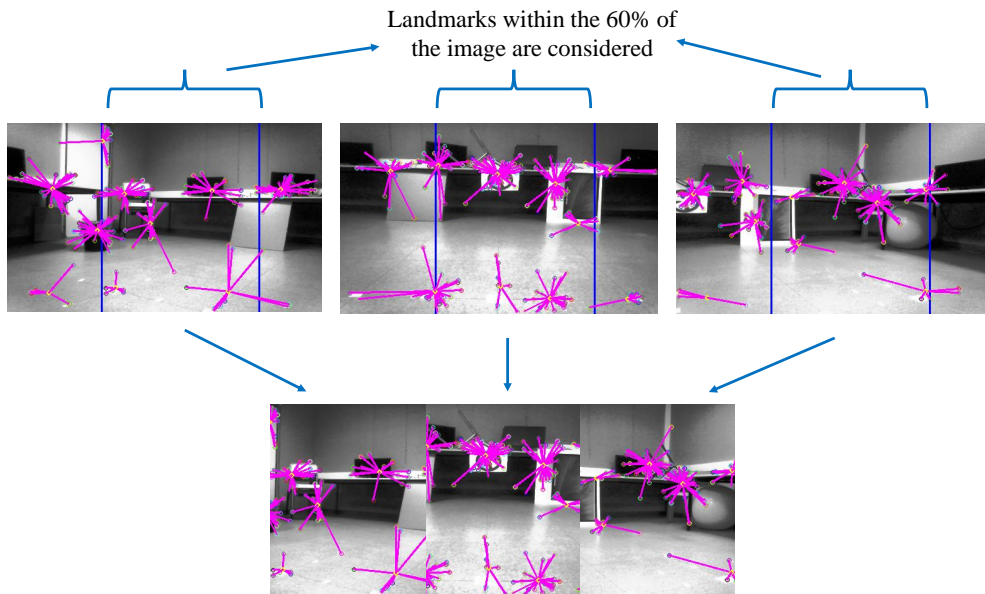


Figure 6.22: Representation of the images considered for the process. Since each image overlaps with the previous and next images, a filter cutting the overlapping borders was used. Therefore, only the landmarks detected within the 60 % of the center of the image were considered.

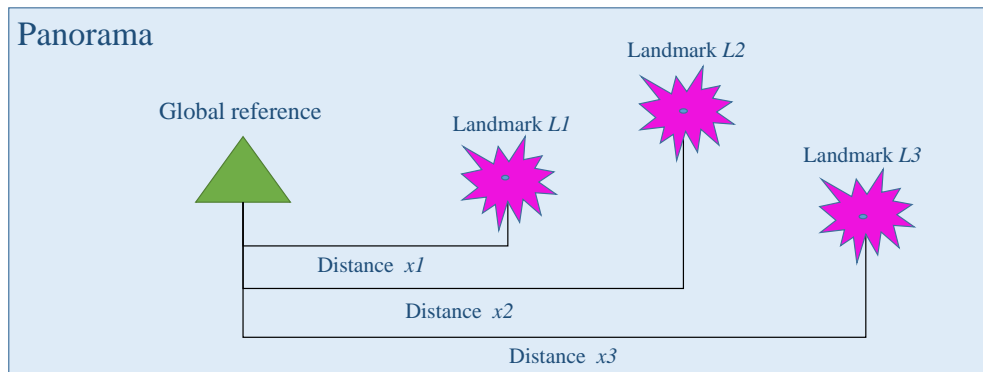


Figure 6.23: Computation of the position of the landmarks. It is calculated with respect to a global reference point (here an artificial landmark has been used). The computation is performed by only considering the *x-coordinates* of both landmark and reference point.

their recognition rate can be computed simultaneously by placing the robot at different places at a time. The learning of the places in both cases is considered to be done in a batch way.

Conversely, each place can be learned and tested it as the robot navigates. We call this,

incremental learning.

The following tests were performed in order to compute the recognition rate by considering the two above types of learning and by comparing them. However, before getting into that, a series of tests were performed in order to find the most adequate SIFT parameters allowing to give a good recognition rate.

We will see that the vigilance term employed in this work to decide if a place has been recognized or not, varies depending on the prior knowledge. For instance, when a single place is learned in advance and then tested at different places, the vigilance term is quite low (here 0.35). On the contrary, when many places are learned beforehand and then tested at different places, the vigilance term would have to increase (here 0.60).

This vigilance term should not be confused to the one used in the *what group* which is used to decide if a *what neuron* has been recognized or not (vigilance term in the *what group* set to 0.95 as mentioned in the place cell recognition layer section).

6.3.3.1 Finding the most suitable parameters

The SIFT algorithm presents 5 different parameters. Depending on the setting values, the description of the image can vary: *nfeatures*, *octaveLayers*, *contrastthreshold*, *edgethreshold*, *sigma*.

In the following tests, we only consider the three last parameters whose description is as follows.

Contrast Threshold: The contrast threshold is used to filter out weak features in semi-uniform (low-contrast) regions. The larger the threshold, the less features are produced by the detector.

Edge threshold: The edge threshold is used to filter out edge-like features. Note that its meaning is different from the Contrast Threshold, *i.e.* the larger the edge threshold, the less features are filtered out (more features are retained).

Sigma: In short, the parameter σ acts as a scaling parameter. For exemple, in the above image, gaussian kernel with low σ gives high value for small corner while gaussian kernel with high σ fits well for larger corner.

Therefore, by tuning the values of these parameters, it would be easy to find the adequate combination that would lead to a good recognition rate.

To this end, the following tests were all conducted on the same two spots as illustrated in figure 6.24 by repeating the same procedure. First, the robot was positioned on place 5 where it could learn the landmark constellation surrounding it as a place. Thereafter, the robot was positioned 50cm to the left of the learned place and the recognition activity was computed.

At each iteration, one or several parameters values were modified.

The first series of tests consisted in modifying only one parameter at a time and checking the recognition rate which maximum value is equal to one (see figure 6.25). The values of the others parameters were the default values of the SIFT algorithm given by OpenCV and remained intact.

$$V_{SIFT} = (nfeatures, octaveLayers, \textbf{contrastthreshold}, \textbf{edgethreshold}, \textbf{sigma})$$

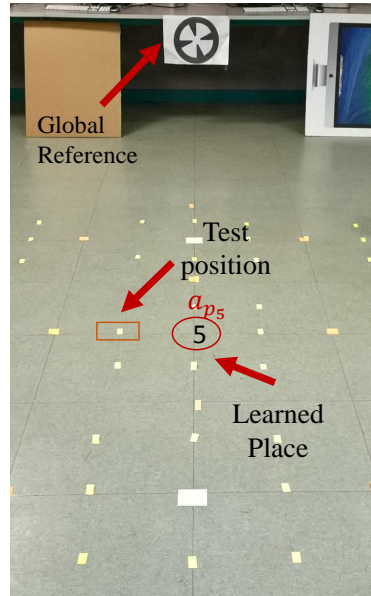


Figure 6.24: Test setting allowing to find the most adequate combination of SIFT parameters that would lead to a good recognition rate. The robot was positioned on place 5 where it could learn the landmark constellation surrounding it as a place. Thereafter, the robot was positioned 50 cm to the left of the learned place and the recognition activity was computed.

$$V_{SIFT} = (0, 3, \mathbf{0.04}, \mathbf{10}, \mathbf{1.6})$$

The resulting values showed clearly that the best recognition rates could be attained with values different from the default ones. This situation was noted for all three parameters where the two highest recognition values are highlighted.

Once these values were known, we wanted to see if by combining them, it was possible to get an even better result. Thus, the recognition rate was computed with the two following combination of the best values (see figure 6.26).

From the results of the first table, it was possible to note that when both *contrast threshold* and *sigma* values are high, the number of outputted key-points was smaller than the number of landmark clusters which prevented the algorithm to run correctly. Conversely, combining one high value (*contrast threshold*) with a low value (*sigma*) shows a good result as it can be seen in the second table. However, it was not better than the results outputted from the above individual tests. The results being not so promising, we did not go further in the combination test.

Conversely, the impact generated by the relation between these two parameters lead us to perform other tests, which would allow us to find a suitable *contrast threshold-Sigma pair value*. As far as the *edge threshold* is concerned, we decided to set it to its default value (10). This decision was based on the fact that the two best recognition rates resulted from two distant values (2 and 20) whose average is almost 10. However, once the best contrast

Contrast threshold	Recognition Rate % (a_{p_5})	Edge threshold	Recognition Rate % (a_{p_5})	Sigma	Recognition Rate % (a_{p_5})
0.008	0.024	0	0.280	0.6	0.157
0.02	0.132	2	0.28	1	0.178
0.04	0.228	4	0.230	2	0.262
0.05	0.246	6	0.274	2.6	0.273
0.06	0.223	8	0.247	3.2	0.341
0.07	0.255	10	0.229	4.8	0.366
0.08	0.279	12	0.231	6.4	0.346
0.1	0.351	14	0.247	8	0.341
0.12	0.400	16	0.247	9.6	0.391
0.14	0.414	20	0.278	11.20	0.293

Figure 6.25: Recognition rate % of place 5 (a_{p_5}) when the robot is placed 50 cm away from it. The values of the SIFT parameters (contrast threshold, edge threshold and sigma) allowing a good description of images are modified one at a time.

Contrast threshold	Edge threshold	Sigma	Recognition Rate % (a_{p_5})
0.14	2	9.6	NA

Contrast threshold	Edge threshold	Sigma	Recognition Rate % (a_{p_5})
0.12	20	4.8	0.378

NA: Number of keypoints not significant.

Therefore, both learning and recognition phases cannot be executed.

Figure 6.26: Computation of the recognition rate % of place 5 (a_{p_5}) by combining the parameters values that gave the highest recognition rates in the first set of results.

threshold-Sigma pair value will be found, we would come back to these two values to perform a comparison.

Consequently, the sigma parameter was set to the two highest resulting values (**9.6** and **4.8**) found in the first set of tests and for each sigma value; the contrast threshold parameter was modified and the recognition rate computed as illustrated in figure (6.27).

From the results, it is possible to confirm the previously given hypothesis about the relationship between both parameters. For instance, left table shows a good recognition rate when the *sigma* value is high (9.6) and the *contrast threshold value* (0.07) is average-low. As for the highest values (0.12 and 0.14), the number of key-points was inferior to the number of clusters preventing to compute the recognition rate. On the other hand, the right table

Sigma= 9.6		Sigma= 4.8	
Contrast threshold	Recognition Rate % (a_{p_5})	Contrast threshold	Recognition Rate % (a_{p_5})
0.02	0.294	0.02	0.389
0.04	0.391	0.04	0.366
0.05	0.363	0.05	0.351
0.06	0.439	0.06	0.366
0.07	0.471	0.07	0.355
0.08	0.443	0.1	0.362
0.1	0.399	0.14	0.432
0.12	NA	0.16	0.445
0.14	NA	0.18	0.344

Figure 6.27: Computation of the recognition rate % of place 5 (a_{p_5}) by fixing the sigma parameter to the two values (9.6 and 4.8) giving the highest recognition rate in the first series of tests (6.25 while testing with different contrast threshold values.

shows the best recognition results when a *low value of sigma* (4.8) is combined with *high contrast threshold values* (0.14 and 0.16).

Similarly, the *contrast threshold parameter* was set to the three resulting best values (0.07, 0.14 and 0.16) and for each of them, the sigma values were modified and the recognition rate computed (see figure 6.28).

As a result, for each *contrast threshold value*, a different *sigma value* appeared, allowing a good recognition rate. While, in the first and third table of figure 6.28, the *contrast threshold-sigma pair value* correspond to the same set of pairs previously found (*sigma=9.6, contrast threshold = 0.07*) and (*sigma =4.8, contrast threshold =0.16*), the second table shows a new value of sigma resulting in an additional contrast threshold-sigma pair value (*sigma=6.4, contrast threshold = 0.14*).

These results seemed to be quite convincing. However, in order to ensure the performance of all these parameters under these values, the remaining *edge threshold parameter* needed to be taken into account.

To this end, we decided to compare the recognition rate of each set of *contrast threshold-sigma pair* by setting the value of the *edge threshold* parameter to each of the two values that gave the best results in the previously presented individual test (see table in figure 6.25). Figure 6.28 illustrates such comparison.

From the results, it can be concluded that the default value of the *edge threshold (10)* gave the best recognition rate. Consequently, the following test were conducted with this edge value.

Once the values of the three parameters were set as potentially giving the best recognition result, a final test needed to be done.

Contrast threshold= 0.07		Contrast threshold= 0.14		Contrast threshold= 0.16	
Sigma	Recognition Rate % (a_{p_5})	Sigma	Recognition Rate % (a_{p_5})	Sigma	Recognition Rate % (a_{p_5})
0.6	0.222	0.6	0.290	0.6	0.247
1	0.239	1	0.276	1	0.354
2	0.279	2	0.417	2	0.340
3.2	0.318	3.2	0.410	3.2	0.342
4.8	0.355	4.8	0.432	4.8	0.445
6.4	0.397	6.4	0.503	6.4	NA
8	0.342	8	NA	8	NA
9.6	0.471	9.6	NA	9.6	NA
11.2	0.330	11.2	NA	11.2	NA

Figure 6.28: Computation of the recognition rate % of place 5 (a_{p_5}) by fixing the contrast threshold parameter to the three values giving the highest recognition rates in the previous tests(0.07, 0.14 and 0.16) while testing with different sigma values.

Contrast threshold= 0.07 Sigma=9.6		Contrast threshold= 0.14 Sigma=6.4		Contrast threshold= 0.16 Sigma=4.8	
Edge	Recognition Rate % (a_{p_5})	Edge	Recognition Rate % (a_{p_5})	Edge	Recognition Rate % (a_{p_5})
10	0.471	10	0.503	10	0.445
20	0.376	20	0.403	20	0.447
2	NA	2	NA	2	NA

Figure 6.29: Comparaision of the recognition rate of each set of contrast threshold-sigma pair by setting the value of the edge threshold parameter to each of the two values that gave the best results in the previously presented individual test.

In fact, since the above tests presented so far were conducted by positioning the robot at only one spot away from the learned place, it is necessary to test several and different spots surrounding the learned place. This would let us not only to generalize the recognition system but also to have a wider perspective of the results allowing to choose the best among the three set of parameters.

To this end, the robot was positioned again at the same place 5 to learn it and then it was positioned at each of the 12 spots surrounding the place, once at a time.

Figure 6.30 illustrates the learned place and the surrounding spots. S_L is a set composing the position of the learned place, S_T is the set composed of the tested places in terms of their

relative position according to a \mathbf{x}, \mathbf{y} coordinate system as illustrated in figure 6.30.

$$S_L = \{(0, 0)\}$$

$$S_{T1} = \{(-1, 1), (0, 1), (1, 1), (-1, -1), (0, -1), (1, -1), (-1, 0), (0, 0), (1, 0)\}$$

$$S_{T2} = \{(0, 2), (0, -2), (-2, 0), (2, 0)\}$$

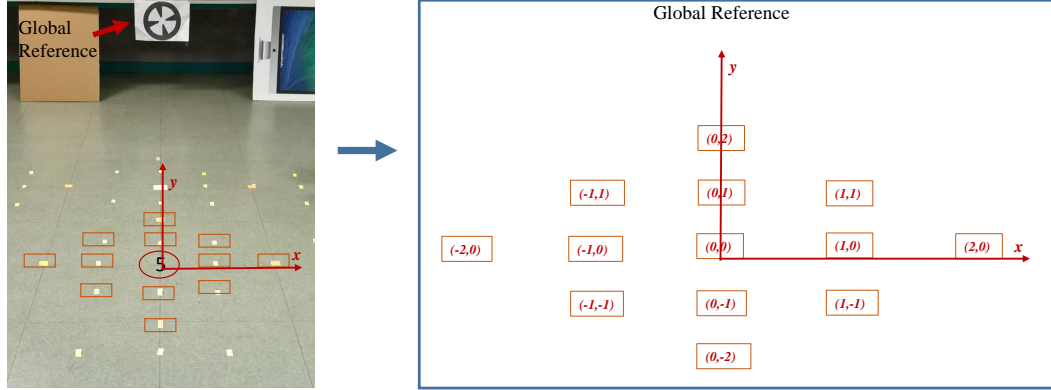


Figure 6.30: Representation of the environment test illustrating the learned place and the surrounding spots where the robot was placed to test and compute the recognition rate in % of place «5» corresponding to $a_{p(0,0)}$.

For a better understanding, the results are presented by following the same topology given by the environment test as shown in the figure 6.31.

The overall results show a recognition rate similar to the ones found when testing only one place. This is good news, as it allows us to maintain a stability of recognition around a given learned place.

The values which have been highlighted are those with the maximum recognition rate at each place among the three test. It can be noted that the table on the top (***Contrast threshold= 0.14, edge threshold=10, sigma =4.8***) has the more number of maximum values than the other tables.

Therefore, this set of values have been chosen in this work to be the most suitable one for performing learning and recognition of a place. They are thus, the baseline of the following tests.

Moreover, the resulting values, led us also to set the vigilance term. Since, we want the surroundings of the learned place (within a ratio of 50 cm) to be considered as the same place, the ***vigilance term was set to 0.35***. Thus, any place which recognition value is superior to such vigilance value should be considered as being the same place.

We are aware that the recognition rate of a same place should be higher than that, and this can be optimized by tuning other parameters such as the number of clusters in the vocabulary of the bag-of-words, the number of landmarks per image or even the lateral

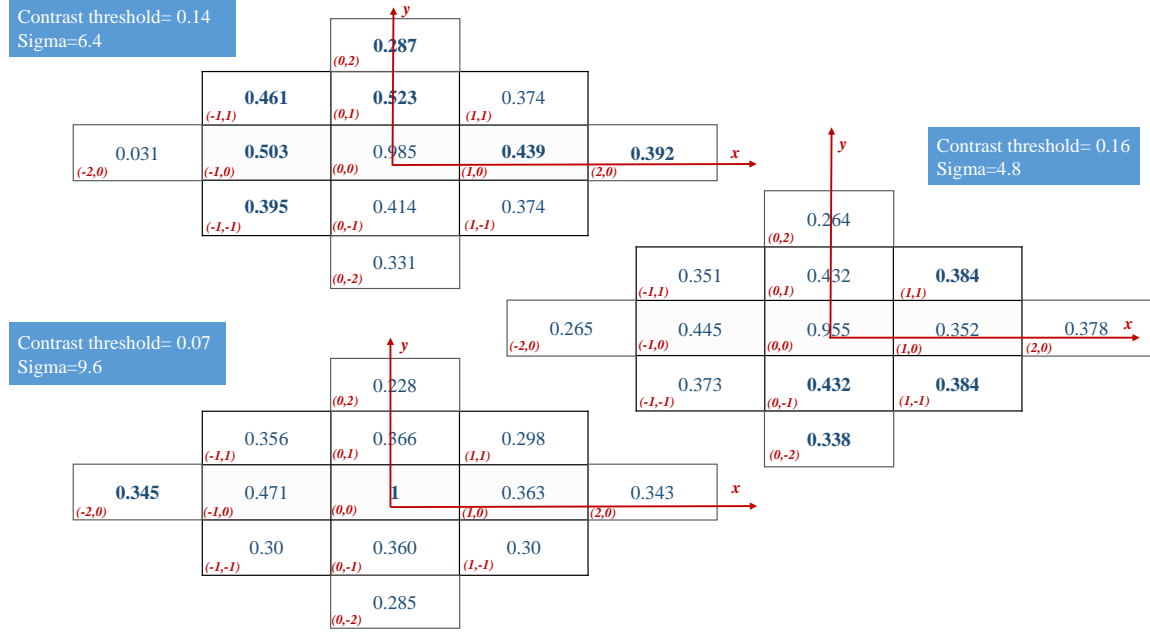


Figure 6.31: Results of the recognition rate in % of place $a_{p(0,0)}$ when placing the robot at different test places (by following the same topology given by the environment test) computed with three different values of sigma-contrast threshold pair. The values which have been highlighted are those with the maximum recognition rate at each place among the three test.

diffusion performed in the where group of the neural structure. However, we consider that these results are rather good and appropriated to continue the rest of the tests (batch learning and incremental learning) to verify the feasibility of the proposed approach.

6.3.3.2 Batch learning

Learning a single place and recognizing it from different places:

A single place in the environment was chosen to be learned and tested. The goal of this test consists in proving if an already learned place, could easily be recognized when the robot was positioned not only at the same place but also at any other place within the environment test.

To this end, the robot was positioned at a place of the test environment from which it could obtain the surrounding images and learn the set of landmarks composing the panorama.

Then, the robot was taken away and placed on each of the nine positions of the test environment including the learned one. At each place, the robot obtained the surrounding images and compared the set of landmarks composing the panorama to the set of learned landmarks and the recognition rate was computed.

The test was performed for each of the following places: {2, 4, 5, 6 and 8} as illustrated in figure 6.32. The following tables and figures illustrate the results.

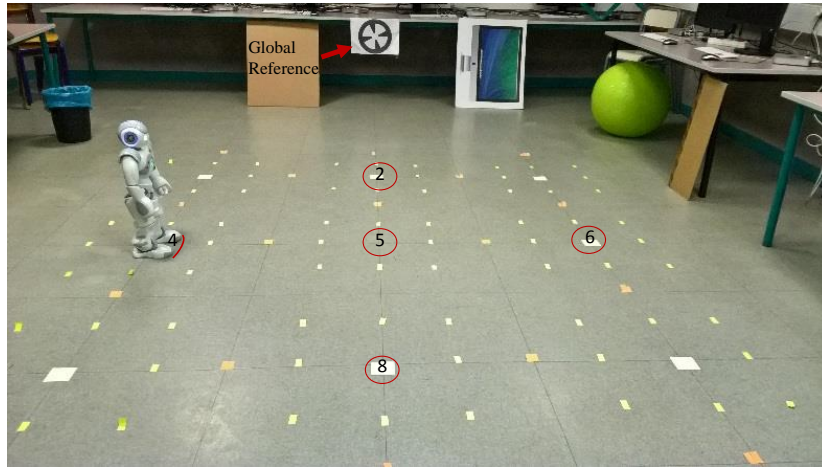
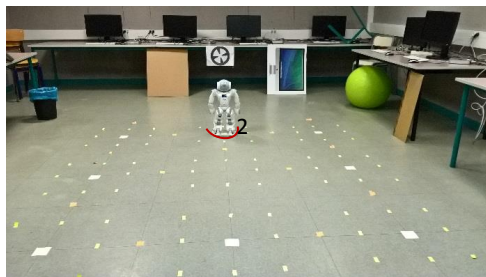


Figure 6.32: Test environment illustrating the different places at which the robot was placed. $\{2, 4, 5, 6 \text{ and } 8\}$

$$S_L = \{2\}$$

$$S_{T2} = \{2, 4, 5, 6, 8\}$$



	TEST #1	
	Learned place Test place	Recognition rate % (a_{p_2})
TEST #1.1	2	0.96
TEST #1.2	4	0.27
TEST #1.3	5	0.32
TEST #1.4	6	0.26
TEST #1.5	8	0.25

Figure 6.33: First set of batch learning tests. The robot learned the place 2. Then for each of the different places $\{2, 4, 5, 6 \text{ and } 8\}$ its recognition rate was computed a_{p_2} .

$$S_L = \{4\}$$

$$S_{T2} = \{2, 4, 5, 6, 8\}$$

TEST #2		
	Learned place	Recognition rate % (a_{p_4})
Test place		
TEST #2.1	2	0.31
TEST #2.2	4	0.98
TEST #2.3	5	0.29
TEST #2.4	6	0.24
TEST #2.5	8	0.27

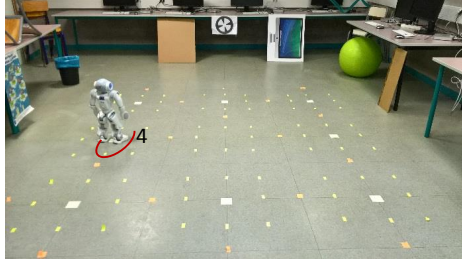


Figure 6.34: Second set of bach learning tests. The robot learned the place 4. Then for each of the different places $\{2, 4, 5, 6$ and $8\}$ its recognition rate was computed a_{p_4} .

$$S_L = \{5\}$$

$$S_{T2} = \{2, 4, 5, 6, 8\}$$

TEST #3		
	Learned place	Recognition rate % (a_{p_5})
Test place		
TEST #3.1	2	0.37
TEST #3.2	4	0.31
TEST #3.3	5	0.98
TEST #3.4	6	0.30
TEST #3.5	8	0.34

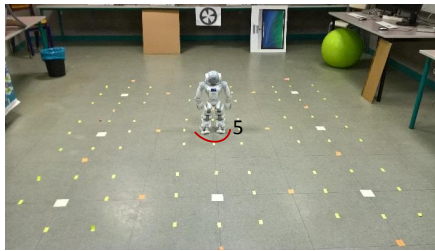


Figure 6.35: Third set of bach learning tests. The robot learned the place 5. Then for each of the different places $\{2, 4, 5, 6$ and $8\}$ its recognition rate was computed a_{p_5} .

$$S_L = \{6\}$$

$$S_{T2} = \{2, 4, 5, 6, 8\}$$

$$S_L = \{8\}$$

TEST #4		
	Learned place Test place	Recognition rate % (a_{p_6})
TEST #4.1	2	0.32
TEST #4.2	4	0.26
TEST #4.3	5	0.32
TEST #4.4	6	0.96
TEST #4.5	8	0.35

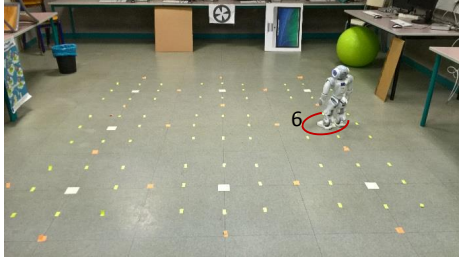


Figure 6.36: Fourth set of bach learning tests. The robot learned the place 6. Then for each of the different places $\{2, 4, 5, 6 \text{ and } 8\}$ its recognition rate was computed $a_{p(6)}$.

$$S_{T2} = \{2, 4, 5, 6, 8\}$$

TEST #5		
	Learned place Test place	Recognition rate % (a_{p_8})
TEST #5.1	2	0.28
TEST #5.2	4	0.30
TEST #5.3	5	0.32
TEST #5.4	6	0.27
TEST #5.5	8	0.97

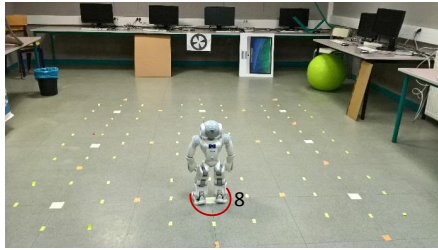


Figure 6.37: Fifth set of bach learning tests. The robot learned the place 8. Then for each of the different places $\{2, 4, 5, 6 \text{ and } 8\}$ its recognition rate was computed $a_{p(8)}$.

When the robot was at the exact same place where it had learned it, the corresponding **place cell** was activated at the highest possible value, whereas as it went away from the place, the recognition value decreased monotonously to the distance, which validates the test. Moreover, it should be noted that the recognition rate of all test places are inferior the *vigilance term* set in the previous test (0.35). This confirms that each of the different places needed to be learned since distance among each of them is of 2m.

Learning all places in the environment and computing their recognition rate by placing the robot at each of the learned places at a time:

Contrary to the above test, this one aims at verifying simultaneously the recognition rate of all places after having all been learned. To this end, all places were first learned by positioning the robot at each one of them. For each place, the set of perceived landmarks together with their relative position was learned. The learning of the landmarks in the *what group* is here done incrementally, which means that every time the perceive landmark is compared to the previously learned. If both happened to be similar, then the landmark is associated with the corresponding neuron and the recognition activity, otherwise a new landmark is learned.

Thereafter, after having learned all places, the robot was positioned again at each of the learned places at a time and the recognition activity of all places was computed simultaneously.

The table presented in figure 6.38 shows the resulting recognition rate of all learned places when the robot was positioned at each place at a time.

TEST #6

	Learned places Test place	Recognition rate % (a_{p2})	Recognition rate % (a_{p4})	Recognition rate % (a_{p5})	Recognition rate % (a_{p6})	Recognition rate % (a_{p8})
TEST #6.1	2	0.96	0.79	0.81	0.74	0.77
TEST #6.2	4	0.75	0.99	0.83	0.71	0.75
TEST #6.3	5	0.29	0.50	1	0.73	0.82
TEST #6.4	6	0.22	0.44	0.61	1	0.72
TEST #6.5	8	0.24	0.49	0.70	0.66	0.98

Figure 6.38: Tests performing learning all places in the environment {2, 4, 5, 6 and 8} and computing their recognition rate by placing the robot at each of the learned places at a time.

Since all places were previously learned, during the test phase all of them were activated regardless the robot test position. However, while the activity of the place corresponding to the current robot test position got the maximum value of recognition, all the others were activated with a smaller recognition value.

Figure 6.39 shows the recognition activity of all places when the robot was positioned at place 4 (see TEST #6.2.).

It should be noted that when testing a single place previously explained, it was necessary to test the robot position at each place for every learned place (a total of 25 tests giving each of them a recognition value).

Conversely, in this test all places were directly activated by only testing one of them (a

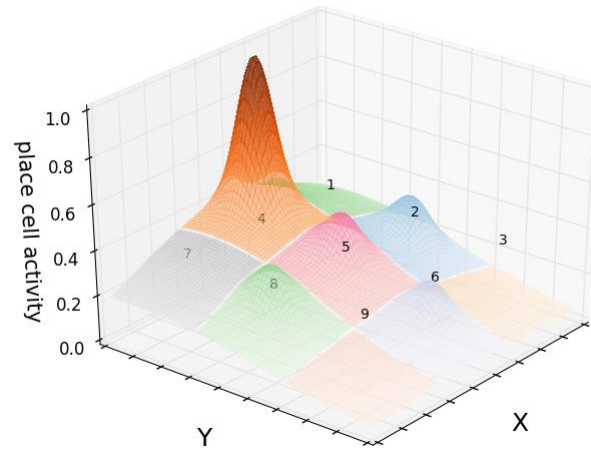


Figure 6.39: Recognition activity of all places when the robot was positioned at place 4 (see TEST #6.2.).

total of 5 tests giving each of them, 5 recognition values). Additionally, the overall recognition values are higher than the ones presented in the previous test. For instance, if we look at the table those resulting when the robot was positioned at place 4, the adjacent values do not fall below **0.61**. This is due to the fact that the system has in its memory more information as it has learned several places before the testing phase took place. In the first test case instead, the recognition was based on the prior knowledge of **only one place**.

Therefore, the vigilance term set in the first case at 0.35 can be and should be increased when the prior knowledge increases.

6.3.3.3 Incremental learning

This test consists in verifying if the robot can learn new places as it navigates within the environment. The idea of it is to learn a place every time the robot sees it for the first time or every time it does not recognize it when it compares it to others previously learned. The decision of knowing if a certain amount of recognized landmarks is enough to state that the place is known or not, is given by a **vigilance term** similar to the one described in the ART model. After having concluded in the last test that the vigilance term increases with the quantity of information learned, the vigilance term used in this test should be set to a higher value. In the meantime, we decided to fix it to **0.61** based on the latest results.

When the robot starts exploring the environment for the first time (starting point in a new environment), it is sure that the very first place it encounters will be unknown. Therefore, it keeps in memory all landmarks and their relative position that it perceives in its surrounded environment (360°) and learns the resulting landmark constellation as a new place.

When a place is not recognized, the current perceived landmarks together with their positions are learned. The recognition process is undertaken by comparing the current perceived

panorama with those previously learned. If the majority or all of the landmarks happened to be similar enough and located at the same or nearby positions, then the place is considered to be recognized and consequently, there is no need to learn. If on the contrary, only few landmarks or none are recognized, the place needs to be learned.

In order to validate this, the robot was first positioned at place 4th (left image in figure 6.40) and from there, it learned the set of landmarks composing the perceived panorama. Thereafter, it was positioned along the line in the next place 5th (middle image in figure 6.40) and the recognition of the first learned place 4th was computed. The same process was done for the 6th place (right image in figure 6.40), where the recognition of both 4th and 5th places were computed.

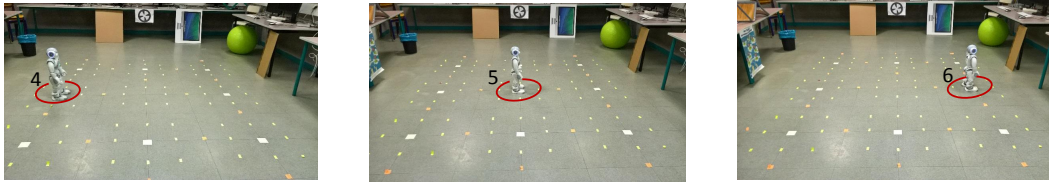


Figure 6.40: Three different places at which the robot was placed. It learned each of them in an incremental way.

The table in figure 6.41 shows the result of the learning sequence as the robot walks along the line covering places 4th, 5th and 6th over the time.

TEST #7					
Learned and Recognition Place	4	5		6	
Temps Test place	t_0	t_1	t_1'	t_2	t_2'
4	1_{Learned}	0.26	0.35	0.23	0.29
5	0	0	1_{Learned}	0.56	0.72
6	0	0	0	0	1_{Learned}

Figure 6.41: Recognition rate results of the learning sequence as the robot walk along the line covering places 4th, 5th and 6th over the time.

At the beginning, when the robot is at place 4th at time t_0 , the robot learns the place

as a new one (maximum value equals to one). Since no other place had been learned before, the values of the others are nil or not considered.

Then when the robot is at place 5^{th} at time t_1 , the recognition process is triggered to compute the value of place 4^{th} . However, since the recognition value is lower than the **vigilance term** ($0.26 < 0.61$), the robot decides to learn it immediately after at t'_1 . Then, the same situation is repeated when the robot is place 6^{th} at time t_2 . Since the recognition values of both previously learned places are inferior to the vigilance term ($0.23 > 0.61$ and $0.56 < 0.61$), the robot learns the place at time t'_2 . As the number of places increase, the recognition rates of the adjacent places also increase.

Figure 6.42 illustrates the activity of the three places over time.

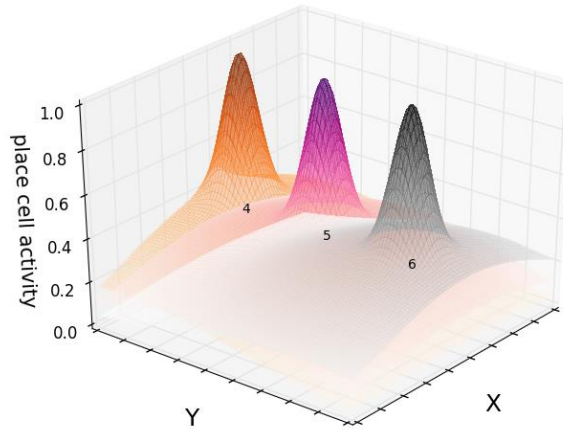


Figure 6.42: Activity of the three learned places over time.

In order to perform the next set of tests requiring the comparison of both type of learning, it was necessary to perform a test in which the learning was incremental while considering the same places tested with the **batch learning test**.

Therefore, the same test was performed, but this time, places 2^{nd} and 8^{th} were additionally considered.

6.3.3.4 Batch vs Incremental learning

This final test consists in seeing the implications of both batch and incremental learning at the recognition phase and highlight the differences.

Consequently, after having learned the ensemble of places: 2, 4, 5, 6, 8 in both, incremental and batch way as explained in the above corresponding tests, the robot was positioned between two adjacent places at each time and the recognition activity was computed.

Figure 6.44 illustrates the test environment with the learned places designed in red circles and the tested places in orange squares. Furthermore, the results table is also designed by following the same topology given by the environment test. The learned places in pink and

TEST #8									
L&R place	2	5		4		8		6	
Temps Test Place	t_0	t_1	t_1'	t_2	t_2'	t_3	t_3'	t_4	t_4'
2	1_Learned	0.32	0.55	0.26	0.37	0.25	0.35	0.23	0.27
5	0	0	1_Learned	0.49	0.68	0.57	0.79	0.49	0.67
4	0	0	0	0	1_Learned	0.45	0.66	0.37	0.43
8	0	0	0	0	0	0	1_Learned	0.45	0.60
6	0	0	0	0	0	0	0	0	1_Learned

Figure 6.43: Recognition rate results of the learning sequence as the robot walks along the line covering places 2nd, 5th, 4th, 8th and 6th over time.

the places where the robot was positionned to test the recognition rates in blue.

For a better understanding, the next results are presented by following the same topology given by the environment test as shown in figure 6.45. The results show the recognition rate of all learned places when the robot is positionned at each of the different tested places: 11, 12 and 10, 13 respectively (see happy smiley in Figures 6.46, 6.46 illustrates).

The ensemble of results shows, that in overall, the recognition rates of all places which were learned in the batch way are higher than those of the same places learned incrementally. This situations was not exactly what we were expecting as result. In fact, the way in which both were coded, made us think that the results could be similar. We can only assume that this is due to the fact that the knowledge in the batch learning is all gathered in advance and is not modified anymore, this gives an stability an accuracy on the recognition task. Conversely, knowledge gathered incrementally moves constantly by increasing as the experiments are done. Further tests, should be conducted to prove this hypothesis.

Despite this, tests gave a correct recognition when the robot was placed at places 11th and 12th as illustrated in figure 6.45. In both cases, the recognition rates of places surrounding the test places were higher than the rest, implying that the robot could be at any of either place (see highlighted values, dark for the second highest and even darker for the highest).

However, something unexpected happened when performing the same test on places 10th and 13th as illustrated in figure 6.46. In fact, they both gave a completely incorrect recognition of the place (see highlighted values, dark for the second highest and even darker for the highest).

Since, such situation was difficult to explain out of these results, another set of tests

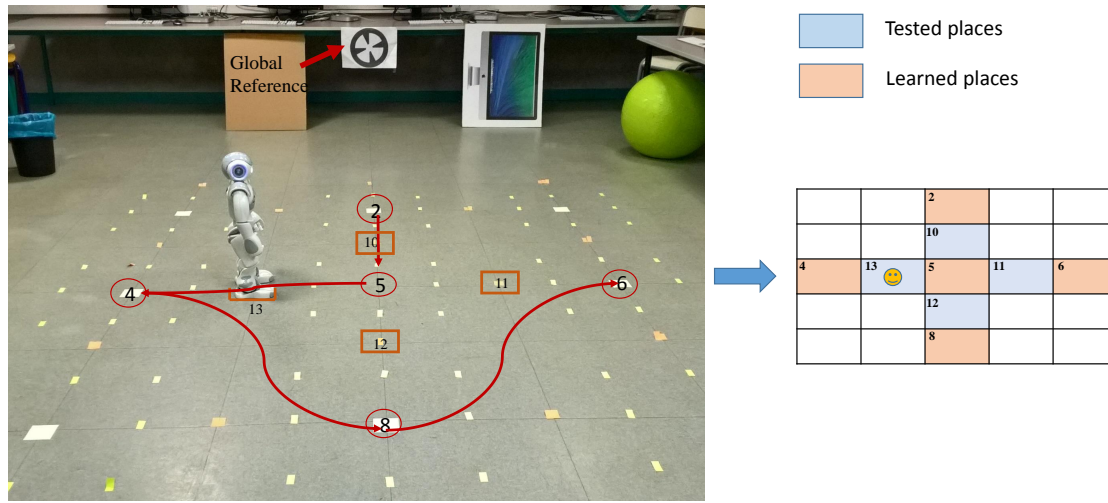


Figure 6.44: Test environment with the learned places designed in red circles and the tested places in orange squares. The red trace represents the path navigated by the robot when learning the places in the corresponding order: {2, 5, 4, 8, 6}. The results table, designed to the right, follows the same topology given by the environment test. The learned places in pink and the places where the robot was positioned to test the recognition rates in blue.

Robot tested position

Tested places

Learned places

TEST # 9 Place 11

		2	0.29			
		10				
4	13	5	0.54	11	6	0.56
		12				
		8	0.47			

TEST # 10 Place 12

		2	0.25			
		10				
4	13	5	0.55	11	6	0.44
		12				
		8	0.45			

Recognition rate % after Batch Learning

		2	0.29			
		10				
4	13	5	0.63	11	6	0.71
		12				
		8	0.68			

		2	0.25			
		10				
4	13	5	0.66	11	6	0.61
		12				
		8	0.67			

Figure 6.45: Recognition rate of all places learned incrementally and in a batch way when the robot is positioned at places 11th and 12th.

needed to be performed.

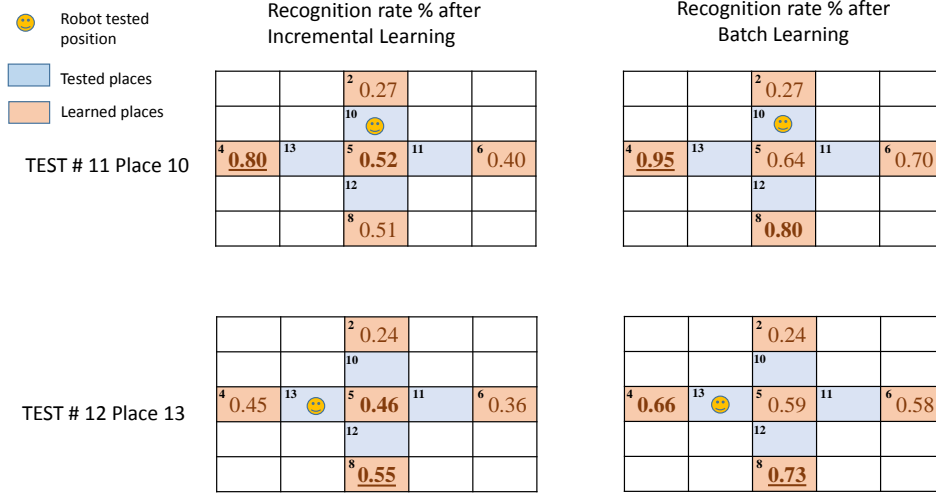


Figure 6.46: Recognition rate of all places learned incrementally and in a batch way when the robot is positioned at places 10th and 13th.

Hence, we decided to perform different test by placing the robot at the same test place. To this end, place 13th was chosen since it gave an incorrect answer in the previous test. The tests consisted in learning only three places in both ways (batch and incremental) but in different order every time.

The expected situation would be that regardless the order of places considered, the recognition values should always be the same. The tables in figure 6.47 show these results.

Unfortunately, the results did not show up as expected. Not only the recognition values vary from test to test, but also the second highest value was not the expected one (tests #13, #14, #15 and #16).

Only tests #17 and #18 corresponding to the learning sequence of places 6,4,5 and 6,5,4 gave accurate results which led us to conclude that if the robot test place (here 13th situated between places 6th and 5th) is closed to the first learned place from the sequence, then the recognition value would be accurate.

After a thorough analysis of this situation, it is assumed that the problem comes from the fact that as new places are learned by the system, the perceived landmarks are first compared to the previously learned. If both are similar based on the high recognition rate, the current landmark is associated with the other one. Hence, at the end of the sequence, most of the places would be rather similar to the very first learned place since their description of landmarks derive from it.

A solution to this problem, would be to avoid the association of two or more landmarks of a given panorama to the same landmark previously learned. In other words, for each panorama, each recognized landmark should be unique.

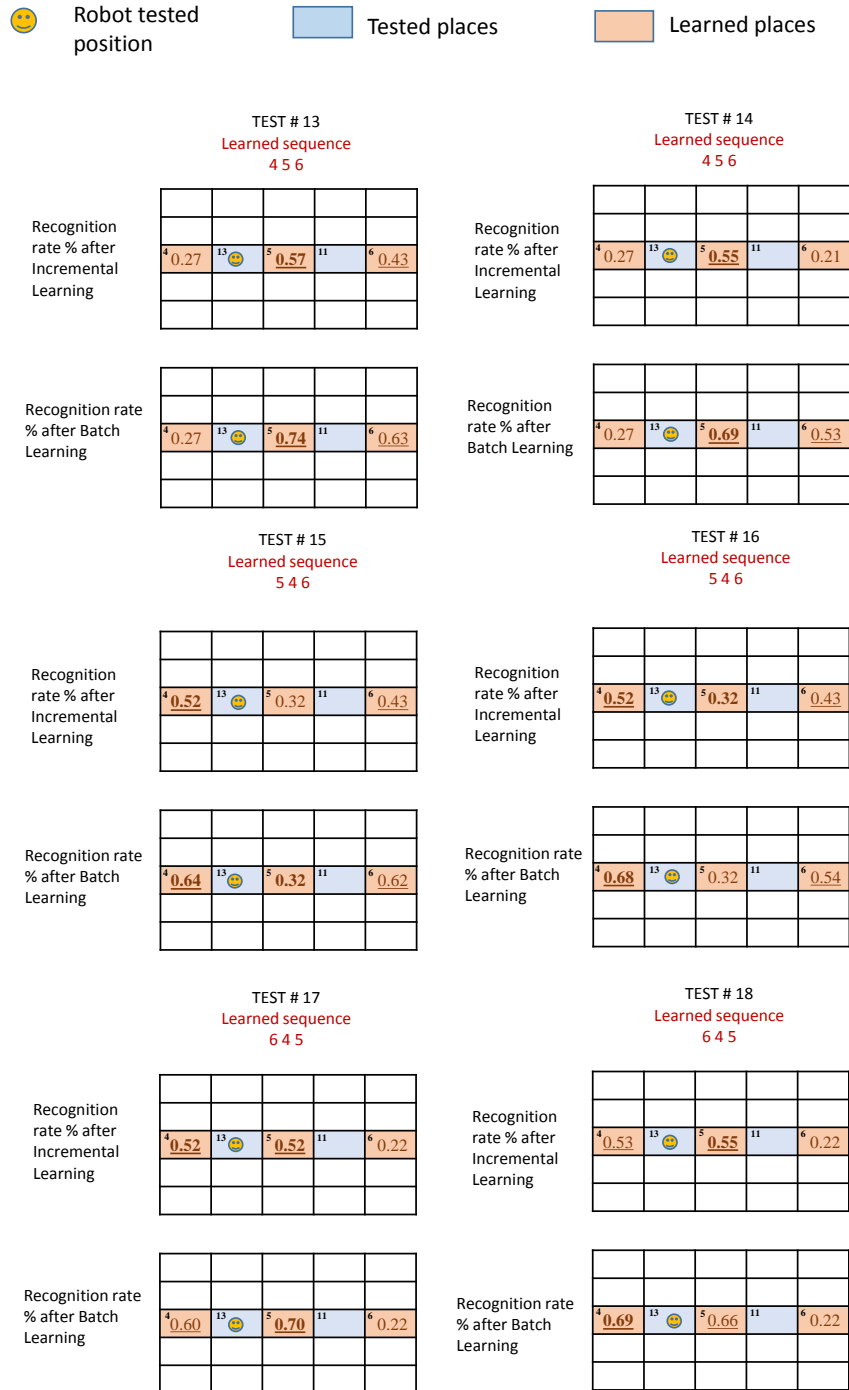


Figure 6.47: Recognition rates of different places learned in different order. Only three places were learned in both ways (batch and incremental) but in different order every time and the recognition rate of all of them was tested while positioning the robot at place 13th.

6.3.4 Discussions

In all tests, when the robot was placed at the exact same location of any of the learned places, the recognition rate was approximately of 98%. Despite this fact, different results showed that the recognition of the place dropped drastically when the robot's position was slightly modified with respect to the tested one, specially when performing a single place batch learning. Since, the recognition of the landmarks was accurate; we supposed that the limitation comes from the determination of other parameters such as the number of clusters in the vocabulary of the bag-of-words, the number of landmarks per image or even the lateral diffusion performed in the *where group* as explained previously. This problem opens new perspectives for the improvement of the proposed algorithm.

From the tests, it was possible to see the advantages of recognizing a place when a vast *a priori* knowledge is considered. That is the case of the batch learning test which learns several places before testing in one place. Similarly, the incremental learning test allowed us to validate the learning as the robot navigated the environment, which is an essential property for an autonomous navigation in order to allow the robot to adapt itself to the environment.

The robustness of this approach lies in the fact that even if one or several patterns characterizing the place are removed or not visibly available anymore, a place can still be recognized. In fact, the ambiguity given by clusters with similar descriptors is dealt by the additional cluster position information which are employed together to learn new places and recognize them during robot navigation.

The validation of the Place Cell and recognition layer through these series of tests allowed us to move forward in the integration of it into the whole system. Therefore, the following section presents a series of tests where the robot navigates within a stochastic environment while adapting itself to the unexpected changes.

6.4 Functional Experiments in real environment

6.4.1 Procedure

The following tests were carried out within the same environmental constraints and conditions of previous experiments presented in Rhizome 1 (see chapter 4) and Rhizome 2 (see chapter 5). This, with the aim of having the possibility of, not only, evaluating the functioning of the whole architecture but also of being able to distinguish the functionality given by Rhizome 3 with respect to Rhizome 1 and to Rhizome 2.

Rhizome 3 integrates all previously described modules into one structure. Therefore, in order to test the overall architecture, it is necessary to execute two different tests under which, the robot navigation performance could be proven while showing a slight difference depending on the use or not of a floor plan.

1. Signs and directions extracted from the floor plan
2. Signs given by a command program

In the first test, the floor plan of the test environment was first shown to the robot before the navigation process took place so that it could extract by itself the sequence of signs. In

the second test, the sequence of sign was directly inserted in the architecture by a command program. Thereafter, the robot was placed at the entry of the test environment from which it could distinctly see the first sign that it was expected to recognize, «A».

In both test scenarios, the sign sequence to be followed was: {A, B, C, D}. However, only the two first signs of the sequence were placed in the test environment (see figure 6.48).

Consequently, the robot was supposed to recognize the signs in the environment and navigate accordingly; and whenever the signs were no longer visibly available, the robot was supposed to walk some distance away and learn a new place as a reference point.

In the first test, since the directional meaning of each sign was provided by the floor plan, the directional movement directing the robot to go far away from the last recognized sign was supposed to be known by robot. Therefore, it was expected that the robot would know where to turn even if the expected sign was not visible and then perform the exploratory movements.

On the contrary, when the directional meaning of the signs was unknown (signs given by a command program), the robot was supposed to perform the exploratory movements together with a turning reflex movement in order to learn the new place some distance away from the last recognized sign.



Figure 6.48: Test navigation environment. Only two signs from the sign sequence are placed in the environment.

6.4.2 Results

The results of each test are given separately but presented in a similar way in order to tell the difference given by the use or not of the floor plan. While navigating the environment, the robot was successfully able to perform the following intended actions.

They are explained according to two different situations:

- When the signs were perceived as expected in the environment;

- When the expected signs were not perceived in the environment.

6.4.2.1 Signs and directions extracted from the floor plan

When the signs were perceived as expected in the environment

- When a sign was detected, it was able to compare it to the corresponding sign from the extracted sequence;
- When the comparison gave a negative result, i.e. the detected sign did not match the expected sign, the robot ignored the detected sign and continued **reflex movements (DDRB layer)** to locate the correct sign;
- When the comparison resulted in a positive, i.e. the detected sign was indeed the expected sign, the robot was able to **perform directly the associated movement given by the map (MDIRB layer)** and **simultaneously learn the association (SRMA)**;
- When a sign was faraway, it was able to **get closer (TARB layer)**;
- When the expected sign was lost from the robot's visual field while approaching it, it was able to perform a **reflex behavior to search for it again (LSSRB layer)**.

When the expected signs were not perceived in the environment

- When the expected sign was not perceived in the environment, after having performed the **associated movement of the previous sign** or a **reflex turning direction movement**, the robot was able to **walk some distance away by performing exploratory movements (WERB)** and **learn a new place (PCLR)** as a reference point;
- Even though the expected sign was not perceived in the environment, the information about the direction to take was known by the robot. Therefore, after having learned the place, it was able to **turn towards the expected direction (MDIRB)** and associated it to the new learned place. This case only happened the first time an expected sign was not perceived;
- In the other cases, the robot was able to perform a **reflex turning direction movement (WERB)** based on the knowledge of the direction of origin and the previously performed movements. This knowledge prevented the robot to go back to previous visited places by directing it towards the exploration of new unvisited ones.

Figure 6.49 shows a summary of the results obtained in the form of the activation of the output neural groups corresponding to the different behaviors performed by the robot, as well as the activation of the reinforcement signal allowing the learning association and the place cell learning behavior, over time. The movements were a result of either the recognition, the proximity or the absence of any signs from the extracted sign sequence. The activities are

explained chronologically and refer to the descriptions and figures of section 4.2.3 in chapter 4, section 5.2.3 in chapter 5 and section 6.2.2 of the present chapter. In each of the (a, t) plots shown, a is the binary activation of each neural group or the reinforcement signal and t the time seconds in terms of a PerAc cycle. For the sake of simplicity, the plot labeled Cumulative target approaching movements combines all movements undertaken in one go by the robot to approach a particular sign using TARB. For more details of this layer, refer to the result section of chapter 4.

$t_0 - t_8$ When the robot recognized **sign A** at time **t_0** , its corresponding neuron in the SRMA layer of the first behavioral module got activated and remained like this while it was still in the robot's field of view (*Dynamic visual perception* plot in figure 6.49). In the meanwhile, it triggered the activation of the neurons in the TARB layer allowing the robot to approach the sign (*Cumulative target approaching movements* plot in figure 6.49) until time **t_8** . At that same time the robot was close enough to the sign to activate the proximity sensor.

$t_8 - t_9$ Since the associated movement to perform was already known by the robot: **right direction** movement associated to **sign A**, the robot turns directly to the right until detecting **sign B** at time **t_9** (*Right turn* plot in figure 6.49). Simultaneously, the reinforcement signal got activated allowing the robot to learn the said association in the SRMA layer of the first behavioral module (*Learning Association* plot in figure 6.49)

$t_9 - t_{17}$ After having recognized **sign B**, the robot performs the corresponding movements in the TARB layer allowing it to approach the said sign (*Cumulative target approaching movements* plot in figure 6.49). The activity of the proximity sensor got a positive value at time **t_{17}** , once the robot was close enough to the sign.

$t_{17} - t_{18}$ Alike sign **A** earlier, current **sign B**'s associated movements was already known by the robot: **left direction** movement associated to **sign B**. Therefore, the robot turned directly to the left. Simultaneously, the reinforcement signal got activated allowing the robot to learn the said association in the SRMA layer of the first behavioral module (*Learning Association* plot in figure 6.49).

$t_{18} - t_{24}$ After having performed the left turn movement, the robot realizes that the next sign is not in its field of view as expected. Therefore, the PC signal gets activated and the robot performs the corresponding exploratory movements in the WERB layer allowing it to walk some distance (until **t_{24}**) away in order to learn a new place (*Exploratory movements* plot in figure 6.49).

$t_{24} - t_{26}$ During this time, the robot detects and extracts the natural landmarks perceived in its surroundings. Then based on their descriptors and relative position, the PCLR layer allows the robot to learn the place (*Place Cell learning* plot in figure 6.49).

$t_{26} - t_{27}$ Even though the expected sign was not visible, the direction associated to it was in the memory of the robot. Therefore, the robot was able to perform the associated movement to the right (*Map Sign Direction* plot in figure 6.49). Moreover, since at the same time, it learned the association between the performed movement and the already learned place (*Learning Association* plot in figure 6.49).

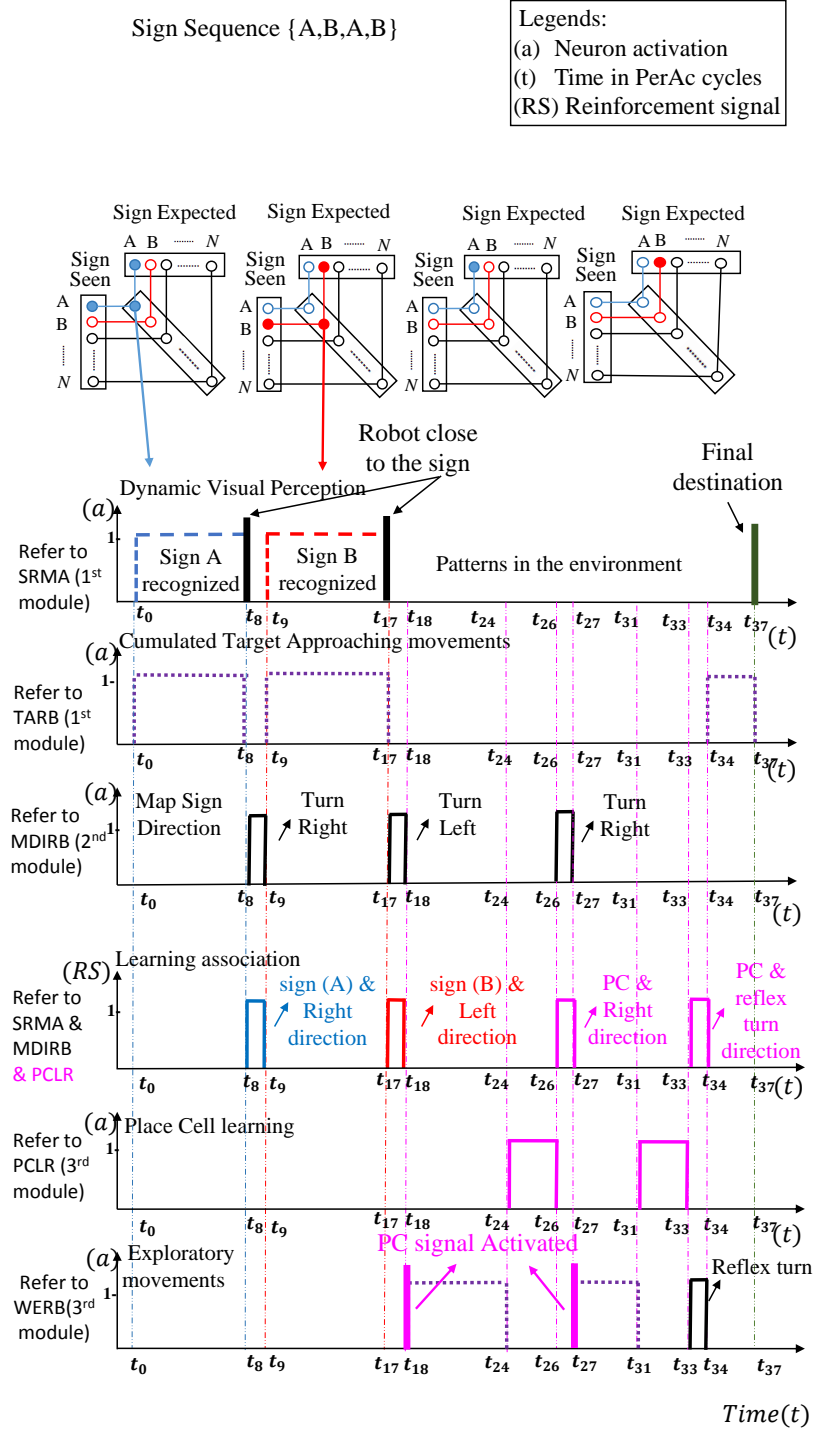


Figure 6.49: Summary of Rhizome 3 results obtained when the signs and directions are extracted from the floor plan. They are obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal (RS) allowing the association learning, over time. In each of the (a, t) plots shown, (a) is the binary activation of each neural group and t the (t) time seconds in terms of a PerAc cycle.

$t_{27} - t_{31}$ After having performed the right turn movement, the robot realizes once again that no sign is in its field of view as expected. Therefore, the PC signal gets activated once again and the robot performs the corresponding exploratory movements in the WERB layer allowing it to walk some distance (until t_{31}) away in order to learn a new place (*Exploratory movements* plot in figure 6.49).

$t_{31} - t_{33}$ Alike the learning place during times $t_{24} - t_{26}$, The robot detects and extracts the natural landmarks perceived in its surroundings and based on their descriptors and relative position, the PCLR layer allows the robot to learn the place (*Place Cell learning* plot in figure 6.49).

$t_{33} - t_{34}$ Once the place has been learned, the robot needs to turn and continue its way. However, since at this point it does not have any information of where to turn, it performs a reflex directional turning movement (*Exploratory Movements* plot in figure 6.49). At that point, it should also perform the related exploratory movements to walk a distance away. However, after having turned it sees at time t_{34} its final destination (an artificial sign in this work).

$t_{34} - t_{37}$ The robot performs the approaching movements directed by the TARB layer of the first behavioral module, in order to get close to the signs referring to the final destination (*Cumulated Target Approaching movements* plot in figure 6.49).

As a result, even though the some of the signs were missed within the environment, the robot was still able to achieve its final destination.

6.4.2.2 Signs given by a command program

When the signs were perceived as expected in the environment

- When a sign was detected, it was able to compare it to the corresponding sign from the extracted sequence;
- When the comparison gave a negative result, i.e. the detected sign did not match the expected sign, the robot ignored the detected sign and continued **reflex movements (DDRB layer)** to locate the correct sign;
- When the comparison resulted in a positive, i.e. the detected sign was indeed the expected sign, and the meaning of the sign was yet unknown, it was able **to perform reflex movements by rotating in one place to search for the next sign and figure out the associated direction (DDRB layer)**. Then learnt it (SRMA layer);
- When a sign was faraway, it was able **to get closer (TARB layer)**;
- When the expected sign was not in its visual field, it was able to perform a **reflex behavior to search for it again (LSSRB layer)**;

When the expected signs were not perceived in the environment:

- When the robot had achieved a complete turn ($\alpha = 360^\circ$) around itself after having performed the **rotatory reflex movement to search for the expected sign (DDRB**

layer), the robot was able to perform a reflex turning direction and walk some distance away by performing exploratory movements (WERB layer). Then, learn a new place as a reference point;

- The robot was able to perform a reflex turning direction movement (WERB layer) based on the knowledge of the direction of origin and the previously performed movements. This knowledge prevented the robot to go back to previous visited places by directing it towards the exploration of new unvisited ones;
- The robot was able to associate the reflex turning direction leading to a new place with the latest perceived sign as well as to the previously learned places (SRMA layer).

Figure 6.50 shows a summary of the results obtained in the form of the activation of the output neural groups corresponding to the different behaviors performed by the robot, as well as the activation of the reinforcement signal allowing the learning association and the place cell learning behavior, over time.

The movements were a result of either the recognition, the proximity or the absence of any signs from the extracted sign sequence. The activities are explained chronologically and refer to the descriptions and figures of section 4.2.3 in chapter 4, section 5.2.3 in chapter 5 and section 6.2.2 of the present chapter. In each of the (a, t) plots shown, a is the binary activation of each neural group or the reinforcement signal and t the time seconds in terms of a PerAc cycle. For the sake of simplicity, the plot labeled Cumulative target approaching movements combines all movements undertaken in one go by the robot to approach a particular sign using TARB. For more details of this layer, refer to the result section of chapter 4.

$t_0 - t_8$ When the robot recognized **sign A** at time t_0 , its corresponding neuron in the SRMA layer got activated and remained like this while it was still in the robot's field of view (*Dynamic visual perception* plot in figure 6.50). In the meanwhile, it triggered the activation of the neurons in the TARB layer allowing the robot to approach the sign (*Cumulative target approaching movements* plot in figure 6.50) until time t_8 . When the robot was close enough to the sign the proximity sensor got activated at time t_8 .

$t_8 - t_{12}$ Since the robot had not associated the current perceived **sign A** with any movement yet, the reflex exploratory action was triggered in the DDRB layer and so was, its corresponding neuron (*Reflex Direction determination Movements* plot in figure 6.50). This continued until **sign B** was detected at time t_{12} .

$t_{12} - t_{20}$ After detection, the total angle of rotation undergone during the reflex movement was computed. The equivalent value (bigger than the threshold value) allowed the association learning of sign **A** with the **right direction** movement triggered by the reinforcement signal **RS** at time t_{12} in SRMA layer (*Learning Association* plot in figure 6.50). While this was taking place, the robot was already performing movements to approach the new sign **B** in the TARB layer (*Cumulative target approaching movements* plot in figure 6.50). The activity of the proximity sensor got a positive value at time t_{20} , once the robot was close enough to the sign.

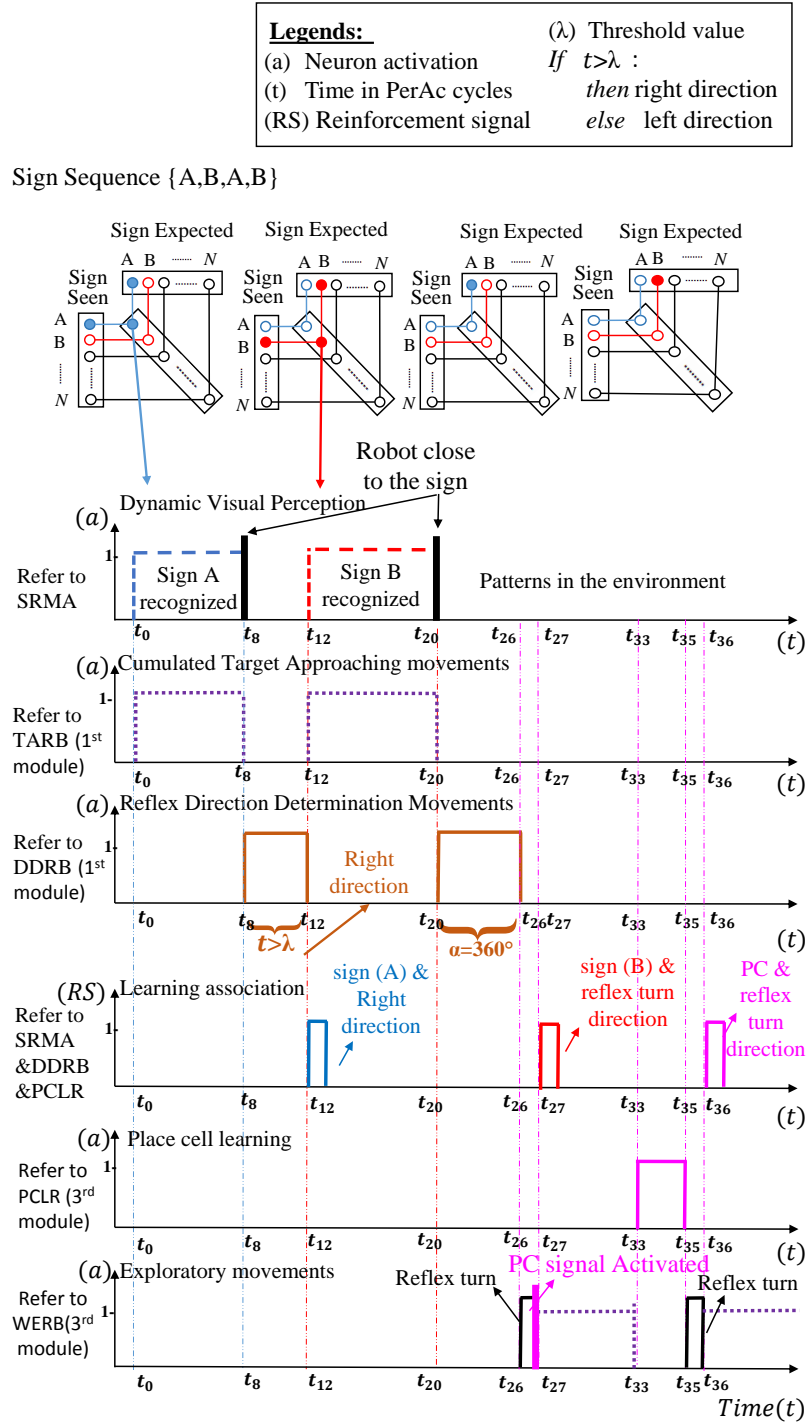


Figure 6.50: Summary of Rhizome 3 results obtained when only the signs are given by a command program. They are obtained in the form of the activation of the output neural groups as well as the activation of the reinforcement signal (RS) allowing the association learning, over time. In each of the (a, t) plots shown, (a) is the binary activation of each neural group and t the (t) time seconds in terms of a PerAc cycle.

$t_{20} - t_{26}$ Alike sign **A** earlier, current sign **B**'s associated movements were not known yet. Therefore, the robot used the DDRB layer actions to locate the next sign from the sign sequence **A** (*Reflex Direction determination Movements* plot in figure 6.50). However, since the next expected sign was not visible at all, the total angle of rotation undertaken achieved a complete turn ($\alpha = 360^\circ$).

$t_{26} - t_{33}$ Since the next sign was not visible as expected, the robot was forced to perform a reflex directional turning movement and walk some distance away from the last recognized sign in order to look for a place to learn as a reference point (*Exploratory Movements* plot in figure 6.50). The behavior was triggered by the activation of the PC signal at time **t_{27}** . Simultaneously, the robot learns the association of the reflex turn direction with the previous sign «**B**» (*Learning Associations* plot in figure 6.50).

$t_{33} - t_{35}$ During this time, the robot detects and extracts the natural landmarks perceived in its surroundings. Then based on their descriptors and relative position, the PCLR layer allows the robot to learn the place (*Place Cell learning* plot in figure 6.50).

$t_{35} - t_{36}$ Once the place has been learned, the robot needs to turn and continue to explore the environment to find the final destination. Since does not have any information of where to turn, it performs a reflex directional turning movement followed by some exploratory movements allowing it to walk a certain distance away (*Exploratory Movements* plot in figure 6.49). Simultaneously, the robot learns the association of the reflex turn direction with the just learned place.

$t_{36} - t_{...}$ After having walked a certain distance, the robot learns once again a new place. The same actions should be repeated until the robot achieves to its final destination.

6.4.3 Discussions

The main idea of executing two different tests was to prove the efficacy of the overall Rhizome 3 architecture at allowing the robot to navigate under the two possible scenarios introduced at the beginning of this dissertation. A deterministic scenario where the *a priori* information is perceived as expected in the environment and a stochastic scenario where some of that prior information is not available any more.

The passage from one scenario to the other was possible to be checked in only one test. However, since the sign sequence could be provided to the robot in two different ways (command program and floor plan analysis), it was necessary to perform two test separately.

Whereas in the first test, the robot had knowledge of the directional meaning of each of the expected signs, in the second test, it had to deduce by itself the movement to perform based on the next expected sign and the learn the resulting association.

In both tests, the robot showed a good performance when the expected sign was not in its field of view. After having realized that the next expected sign was not visibly available, it performed some exploratory movement allowing it to walk farther away and thus learn a new place as a reference point. The learning task was performed in an incremental way. At every point, it learned a new place.

However, both tests differ in the decision of where to go after not seeing the expected sign. In fact, this difference is caused by the availability of the information of the environment

before the navigation activity started. For instance in the first test, the robot could directly turn to the left side after perceiving for the last time one of the expected signs, and turn right after having learned a place towards the same spot where the next expected sign was supposed to be, all thanks to the information of the map.

Conversely, the robot could only realize that the next expected sign was not in its visual field after having achieved a complete 360° turn around itself. Thereafter, a reflex turning movement was trigger based on the previously performed movements, allowing it to turn and then walk towards a new possible place that would probably lead it to the final destination.

If the sequence of signs is small as the one presented in the above tests, using a floor plan in advance, facilitates the accomplishment of the navigation task. The robot can get to its final destinations as illustrated in figure 6.49. However, if the number of missing signs is bigger or if the robot does not have the floor plan in advance, the navigation task can be more complex. The robot would need to explore different places all over the environment in order to finally get to its destination.

In this work, we decided to test until that point, where the robot was able to show a good performance at walking some distances away and learn a place. Moreover, thanks to the return-au-nid layer mentioned at the end of chapter 4, the robot was also capable of returning to the previous learned places and visited signs in order to start a new exploration towards another direction.

Validating the arrival of the robot to the final destination would have required to build an obstacle detection layer in order to prevent the robot to collapse when some unexpected obstacles would appear. However, as it was said at the beginning of this dissertation, that task is out of the scope of this work.

6.5 Conclusion

This chapter presents a hybrid neural-based architecture, Rhizome 3, enabling autonomous robot navigation in complex environments where unforeseen situations are prone to happen. For instance, the occlusion or absence of an expected sign serving as reference in the navigation task.

To this end, a biologically inspired approach for recognizing places within an environment has been presented. It consists of a place cell model allowing place recognition even if one or several patterns characterizing the place are removed or not visible anymore. The recognition process in this work is improved with respect to the state-of-the-art place cell approach. Additionally, the interconnection of the neural groups composing the model is made such that the robot is able to learn new places as it navigates and interacts with the environment to get to its final destination.

Experimental results have validated the advantage of the incremental learning allowing the robot to cope with any unforeseen changes and thus adapting itself to the environment as it navigates within the environment.

Thanks to its generic composition, the place cell model has been integrated to the overall architecture without modifying the functionality or design of the already built-in modules presented in Rhizome 1 and Rhizome 2. Consequently, Rhizome 3 is composed of four

modules: one deliberative and three behavioral modules.

The deliberative module stores the sequence of signs, which is provided to the robot either by a command program or by means of a floor plan analysis allowing the robot to extract it by itself.

The first behavioral module allows the use of the navigation signs as the only *a priori* information available (sequence of signs provided by a commande program). Thus, the robot is able to deduce by itself the movements to perform associated to the signs and learn such association.

The second behavioral module uses the information resulting from the floor plan analysis and directly sends the directional sign meaning to the robot motor output. Therefore, the robot does not need to deduce the movement to perform. The fourth module allows the robot to learn new places by detecting and extracting natural landmarks from the environment. More precisely, it gathers the set of natural landmarks (represented by their unique characteristics and their relative position) that it perceives around itself.

The interconnection of all modules allows the robot to decide on the action to take depending on the provided information. Experimental results integrating the modules altogether, prove the efficacy and robustness of the architecture and robot behaviors when navigating under different and unexpected scenarios. Consequently, the robot is able to detect by itself what it considers to be relevant from the environment as natural landmarks and learns them as a reference point whenever the expected signs are not visibly available in its field of view.

Part III

General Conclusion

Conclusion

Contents

7.1	Summary and Contributions	219
7.2	Short-term Perspectives	224
7.3	Long-term Perspectives	226
7.3.1	Genesis of the Rhizome Architecture	226
7.3.2	Anatomy of the Rhizome Architecture	226

7.1 Summary and Contributions

The work described in this dissertation has been presented as a contribution to the development of autonomous vision-based mobile robot navigation problem, which is a vast ongoing research topic. Thorough the dissertation, several contributions have been presented; from the theoretical point of view to the practical proposition of a new control approach.

Firstly, in order to understand the problematic already mentioned, an introduction (see chapter 1) has been written with the will to guide the reader through the different points, considered important to the author, to achieve a successful robot navigation.

Thereafter, the state-of-the-art vision-based robot navigation strategies has been presented by following a transversal structure proposed by the author in chapter 2. In effect, it is presented in terms of the different functional modules (visual perception, world modelling, localization and path planning) which implication in the navigation task, depends mainly on two type of navigation strategies: mapless navigation and map-based navigation composed itself of map-using and map-building navigation.

The existing relationships among the functional modules and means of integration are defined by a control architecture. Currently, there exists different control paradigms, which, by their applications, characterize the way to solve the robotic navigation problem. In the context of this work, the state-of-the-art of such paradigms has been presented from two points of view in chapter 3.

From the functional point of view, four paradigms can be distinguished: the deliberative approach based on prior knowledge, internal observation of the actions or states of mind. The reactive approach based on a stimulus-response model allowing the robot to deal with very dynamic and unpredictable environments. The hybrid approach, which combines the advantages of both reactive and deliberative approaches, while diminishing their individual

drawbacks. And, finally, the behavior-based approach based on behaviors consisting of a collection of independent behaviors allowing to perform a successful navigation.

From the design point of view, the robotic navigation problem can be tackled by processing the information either in a top-down fashion or in a bottom-up fashion.

Although most of them have shown to succeed at providing a solution to a specific and particular navigation problem, it seems that there is still a long way to go when it comes to respond to different constraints given by a more generic navigation problem.

Therefore, after a thorough study of such approaches, the author came to the conclusion that in order to conceive a control architecture capable of responding to different scenarios constraints in a generic fashion, it was necessary to conciliate all differences found among the state-of-the-art paradigms. Thus, rather than embracing a single approach or following a single path of thought, the author has presented in this thesis, a complete architecture capable of creating a synergy of different approaches by merging them into a neural structure, called **Rhizome**.

The Rhizome architecture was conceived, built and implemented through three different scenarios under which, three interdependent architectures emerged, each responding to different scenario constraints and representing a contribution to the navigation problem. Each architecture has been detailed in the chapters 4 to 6, a summary of which is given below.

RHIZOME 1- Exploring the world with little information

Rhizome 1 is composed of the important modules necessary to allow a robot to reach its final destination with little knowledge of the world in a deterministic scenario. The *a priori* global knowledge of the world represented by a sequence of navigation signs is stored in the deliberative module of the architecture through a command program (see Deliberative module of figure 7.1). Rhizome 1 integrates the sequence into a behavioral module in order to allow the robot to detect and recognize each expected sign during navigation. Moreover, since the meaning of the associated directional implications of the navigation signs is unknown, Rhizome 1 allows the robot to denote them and learn them as a result of a stimulus-response model during navigation. Then, the robot is able to perform movements based on a recall of similar situations and actions previously learned. As a consequence, the behavioral module of the Rhizome 1 architecture differs from the basic PerAc in having a nested PerAc module within its own second level. Hence, it is composed of three layers as illustrated in the behavioral module of figure 7.1.

RHIZOME 2- Autonomous map-based robot navigation

The environment is considered deterministic as in the previous case. The *a priori* global knowledge of the world is gathered by the robot from a paper-based floor plan just prior to real world navigation. Consequently, in an attempt to emulate the cognition process of a human brain when navigating in an unknown building, Rhizome 2 enables the robot to «read» the floor plan of the building, to extract and «memorize» a sequence of navigation signs leading to the final destination and to «recognize» the same signs in the environment

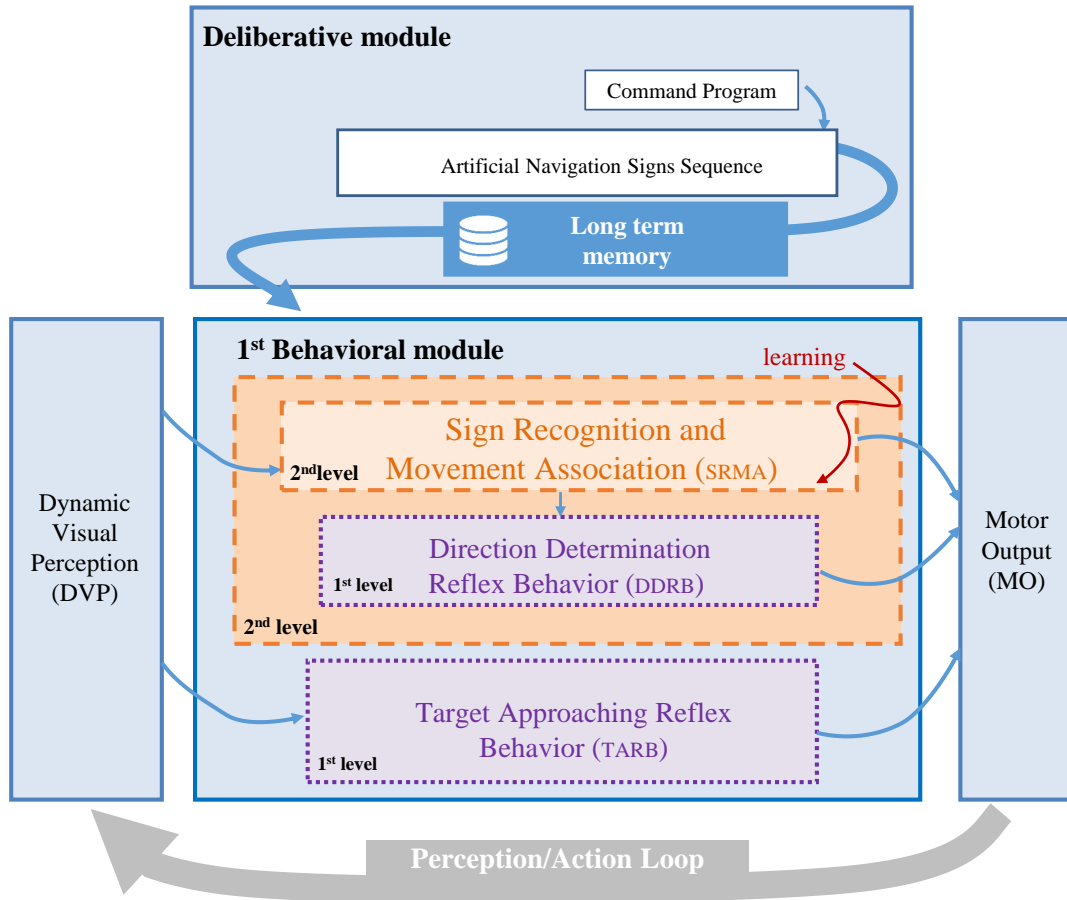


Figure 7.1: Rhizome 1 composed of a basic deliberative module and a behavioral module composed itself of three layers.

while navigating in real time.

The image of the floor plan acquired by the robot camera, undertakes a thorough document analysis process permitting the robot to extract the relevant information for its integration into the system: the **of the navigation signs** together with the corresponding **directional meaning** each sign denotes (see Deliberative module of figure 7.2). Thereafter, based on the provided information, the neural structure allows sign recognition and learning of the association between the recognized sign (see 1st Behavioral module of figure 7.2) and its directional meaning (2nd Behavioral module of figure 7.2) while allowing the robot to perform the corresponding movement directly.

RHIZOME 3- Learning and self-adapting according to unforeseen environment changes

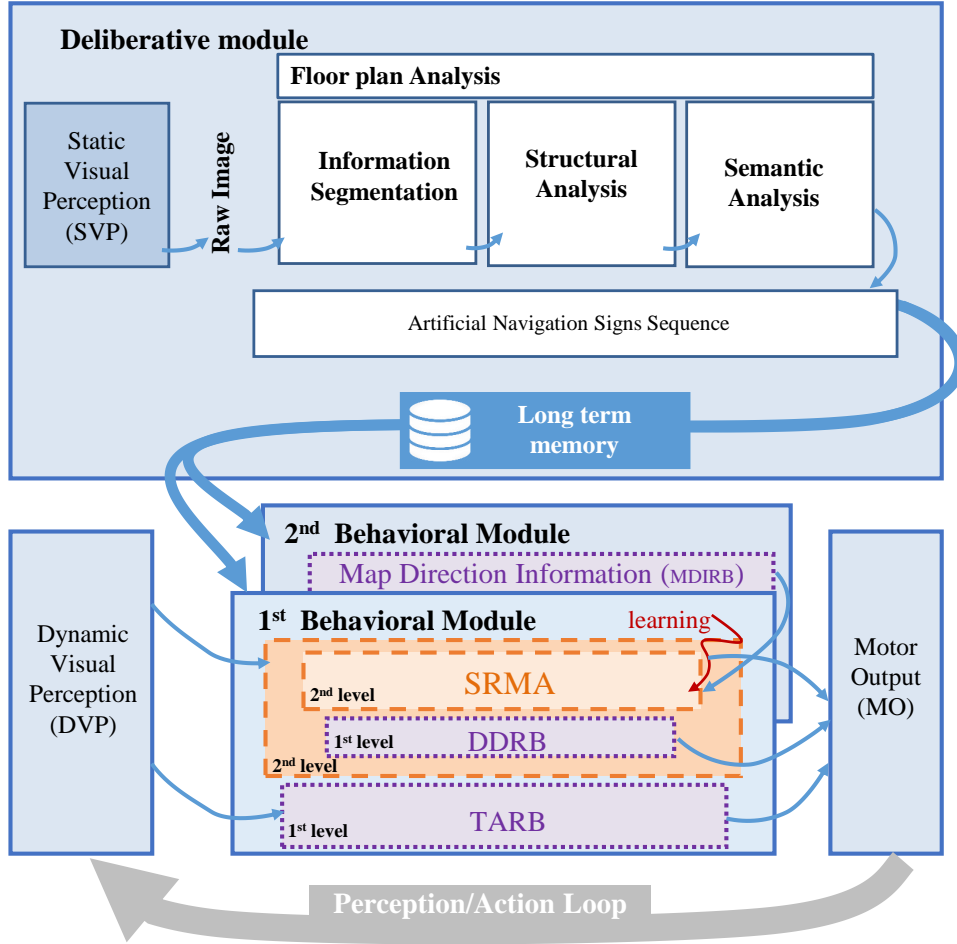


Figure 7.2: Rhizome 2 composed of a deliberative module and two behavioral modules. The deliberative module is in charge of computing a document analysis process on a floor plan. The first behavioral module correspond to the one presented in Rhizome 1 and the second is in charge of process the directional meaning each sign denotes extracted from the floor plan.

The environment is dynamic, stochastic, and thus more realistic. Rhizome 3 is conceived in order to allow the robot to cope with any unforeseen changes involving the occlusion or unavailability of the expected signs. To this end, a place recognition system based on a place cells model is implemented into a third behavioral module. A place is characterized by a set of patterns with their corresponding positions with respect to a given «north» perceived by the robot within its surroundings. The robustness of this approach lies in the fact that even if one or several patterns characterizing the place are removed or not visibly available anymore, a place can still be recognized.

Hence, while the first two behavioral modules are in charge of using the artificial navigation sign sequence stored in the deliberative module to control online navigation, a third behavioral module allows the robot to **find places**, then **learn them** and **recognize them**

as new reference points when the expected signs are not visibly available (see figure 7.3).

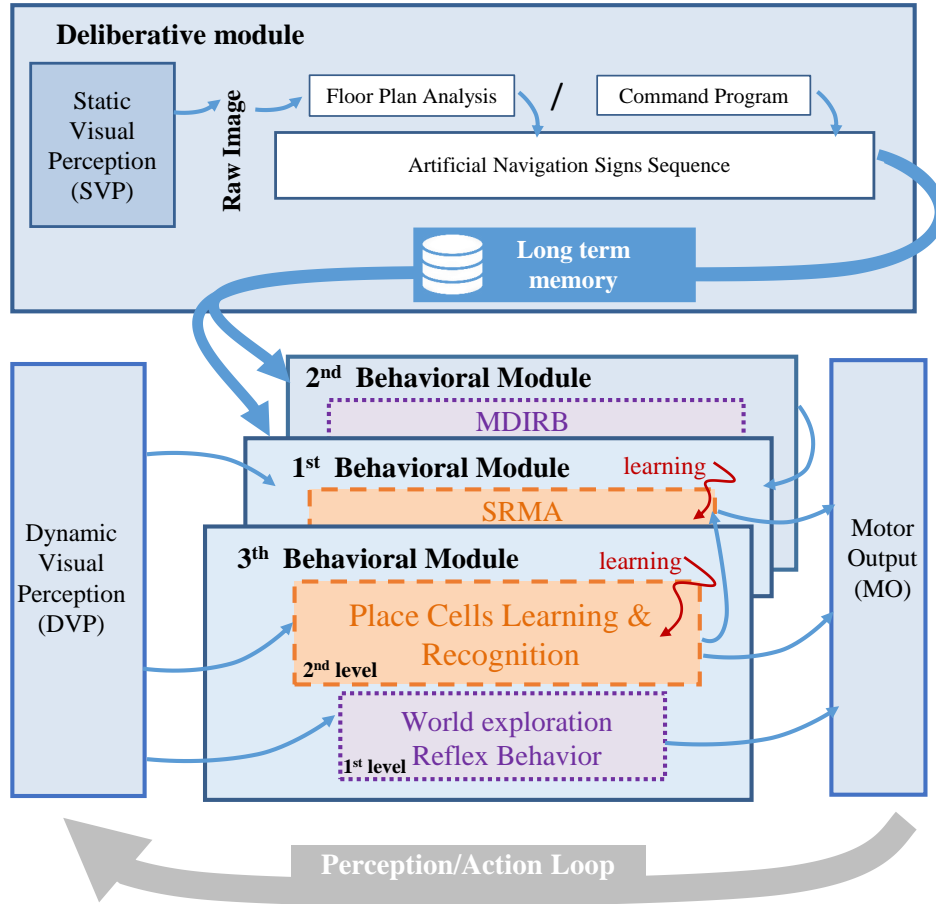


Figure 7.3: Rhizome 3 composed of a deliberative module and three behavioral modules. The deliberative module stores the navigation sign sequence that is either given by a command program or extracted from a floor plan. The Behavioral modules use both *a priori* and dynamic visual information allowing the robot to navigate based on the recognition or absence of the artificial navigation signs. In their absence a place recognition system is performed.

The whole system works in parallel and a '**competitive mechanism**' allows deciding on the best behavior (among the layers and modules) for controlling the robot according to the stimulus received. This is possible because the neural interconnection is done by either excitatory or inhibitory connections allowing or preventing the activation of neurons respectively. Another connection is also used when learning is required: modulation connection. The learning of the associations between the recognized sign and a particular action is conditioned by a reinforcement signal, which represents the internal motivations of the robot.

The architectures should not be regarded as if there was a hierarchy among them or

as if they followed an evolution pattern where each architecture is the improvement of the previous one. Conversely, starting from the simplest scenario imagined, the emergence of the first architecture occurs. Then, its functionality together with its components propels the functionality of a second one by integrating new components, and thus, the entire ensemble of both architectures propel the functionality of the third one.

There is no hierarchy to be considered, each architecture is as important as the others according to its corresponding scenario.

Such imbrication can be better understood by observing the composition of a hypercube, which representation (see figure 7.4) is analogue to the nature of the RHIZOME architecture. While Rhizome 1 can be considered as a square structure, Rhizome 2 composed of the same square and other components forms a cube; and Rhizome 3 composed of new elements forms a hypercube that encompasses Rhizome 2 which itself encompasses Rhizome 1.

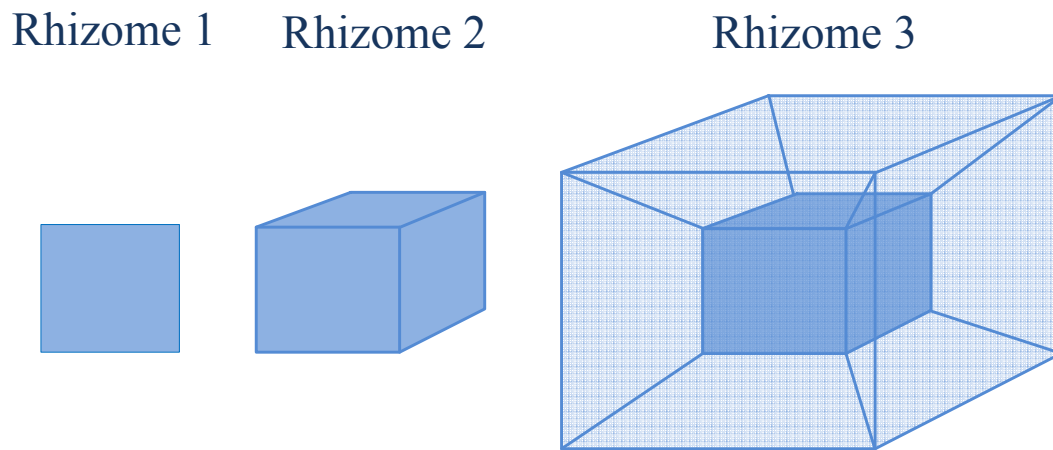


Figure 7.4: Representation of the imbricated set-up of the RHIZOME architecture. While Rhizome 1 can be considered as a square structure, Rhizome 2 composed of this same square, forms a cube and Rhizome 3 a hypercube that encompasses Rhizome 2 which itself encompasses Rhizome 1.

Thus, the anatomy of the Rhizome architecture changes according to the new scenario constrains as the hypercube changes according to its dimension.

7.2 Short-term Perspectives

Allowing a robot to autonomously navigate an environment while being able to switch and self-adapt within an **infinite** number of possible scenarios, means to the author the optimal

goal.

However, given the limited time that PhD work can encompass, it was necessary to first focus on the construction of the foundation of what, as considered by the author, could further grow as a new solution model to the vision-based mobile robot navigation problem. Thus, the lines of work were first traced and followed while bearing in mind the imposed limits.

Consequently, the first bricks of a whole model were assembled by implementing three different architectures functioning under different scenarios and by executing the necessary experiments to validate the proposed approach, as demonstrated in each of the related chapters.

Then, once the efficacy, genericity, robustness, multiplicity and transversality of the proposed architecture was proven, the possibility of building new modules adaptable for further scenarios become clearer. In effect, as previously stated, the generic composition of the architecture allows it to be extended and further developed by adding new modules to the architecture without altering the already in-built layers and modules.

In the short-term, the following ideas, among many others, can be highlighted as future work:

- At the end of the environment navigation under a stochastic and realistic scenario, the robot should be capable of updating the map with the new information given by the place cells recognition. In this way, the map can be built by merging the prior information from the static map and the information related to the new changes due to the emerging behaviors from unforeseen situations. A SLAM technique could be foreseen for this task.
- The performance of the architecture in terms of time speed was neither tested nor compared to the time performance of other existing architectures. Certainly, the author is aware of the importance it represents in any control system, especially in applications where the robot needs to accomplish its task within the given time constraints. Therefore, future work can be led towards the validation of such performance under search and rescue scenarios for instance.
- New scenarios where the signs may be substituted with other relevant visual or non-visual cues requiring the use of different and multiple types of sensors can and should also be considered.
- Finally, a successful implementation of the Rhizome architecture in diverse robotic platforms would allow to prove, confirm, and reaffirm the transversal, generic, hybrid nature of the proposed architecture.

Nevertheless, several points remain in suspense; new roots and shoots coming out to all directions from different nodes; random and unexpected behaviors to be discovered, as the nature of the Rhizome itself dictates it... thereby; leaving open doors to new ideas; to new scenarios, to new passages; to an infinite number of new solutions and possible configurations. Since it is only, in the impossibility of conception and creation of new configurations, solutions, ideas never thought of before in the creator's mind, that the boundaries are imposed.

7.3 Long-term Perspectives

7.3.1 Genesis of the Rhizome Architecture

Rhizome is a philosophical concept developed in the book «A Thousand Plateaus» [1980] by the French authors Gilles Deleuze and Felix Guattari, a philosopher and a psychoanalyst respectively.

The Rhizome term is extracted because it adapts to the concept that guides this present work. Even though the word is used by varying its original meaning, it collects from it, the essential foundation that represents the effort of finding links and relationships among all those theories that are apparently dissimilar but which viewed in the light of their possibilities, they can be connected engulfing themselves within the functioning of a concept.

A comparable orientation can be found in the work of the Austrian-born American physicist Fritjof Capra [1996]. In his book «The Web of Life», Capra synthesized the systems thinking literature into three different but interdependent conceptual dimensions. Pattern of organization, structure, and the life process. The pattern of organization perspective is related to the «autopoiesis». A term coined by Humberto Maturana and Francisco Varela [1980] that refers to a system capable of reproducing and maintaining itself. However, Capra's perspective of living systems is enriched by the notion of Rhizome since the unity of things, according to this concept, is due to the fact of finding roots that unify and link theories with practices. Finding the structure of hidden passages that are not seen to the naked eye; that are underground in somehow; that have to be produced and understood from different theories sometimes related but in most of the cases opposed.

The Rhizome architecture attempts to trace common threads among the different theories and their application to solve practical problems of modern robotics, without assuming a trend. On the contrary, it observes and analyzes all of them, to build a unity among them by appreciating their dissimilarities, their oppositions and by placing them at the service of a robotic operation. The acquisition of deep bonds existing in the various theories also reflects the need to place multiple centers of operation, which in turn, are linked to the general framework in which it operates towards the periphery, towards the outward forms where the practicality of the rhizome concept is perceptible.

7.3.2 Anatomy of the Rhizome Architecture

Transversality in the thought is a creative act. If one finds two ways of seeing things: dissimilar, different, opposite; it would be necessary to plan lines that link them all in order to move forward. Lines that are transversal, labyrinthine, and oblique. Lines that are passageways, tunnels, paths that lay bridges among concrete forms, to which, diverse routes of thought, established as open or close paths, can reach. Lines that seemingly do not have anything to do with other open or close paths; but if they are deeply thought, hidden links that are not revealed immediately to the naked eye can be found. Because they are theories that dispute among themselves the supremacy and the interest of finding practical possibilities to which, engineers and researchers attribute importance. Nevertheless, the technical thinker, cannot enter into these disputes, but instead, accept them as moments or integral parts of a wider

practical-technical thought: Integral parts of a whole that is being constructed and for which such disputes do not have any interest.

These links are underground and it is necessary to construct them, reconstruct them, establish them and re-establish them, by thoroughly studying what really relates them. By abandoning a structured and fixed vision in the path itself. By not choosing a path as unique. By moving from the center to the periphery, from the bottom to the top. By penetrating in the knowledge of the characteristics of each one, without remaining fascinated with what has been found; but instead, by taking a step back to be able to see beyond the investigated path, the studied theory. Consequently, by being able to see other theoretical options and find invisible overlaps that should be established in order to assemble a **unified Rhizome architecture**.

The links among dissimilar theories are not usually discernable to the naked eye, since it is at the surface where the differences are accentuated; and sometimes the contradictions too.

Thus, theorists and researches embrace with fascination one or other current of thought within their proposed discussions. The limits that technicians and engineers take, follow different lines that become more and more engrained depending on certain successful results, which lead them to assume a path that is considered valid above all the others. Thereby, limiting and closing any other possible paths. Denying other possible passages that can be part of a whole. Part of a hybridization in the thought of a Rhizome architect who considers each theory and established trace as a passage itself, as a link. From that, it is said, «*Omnis determinatio est negatio*» [Spinoza, 1804]. Every determination and statement of position of a theory denies the others.

The thought, the method and the Rhizome architecture, can neither deny the existence of the dissimilarity nor the existence of the contradiction, since they hybridize all when looking for, establishing or creating links. Because it is precisely human intelligence creating artificial intelligence that it cannot act denying neither the possibilities nor the contingencies to obtain a whole; an organic and related, dependently and interdependently whole.

It is a thought that does not think the organic as a structure; but instead, as a whole in itself and in each one of its parts. Spherical in its links and relations and unfinished as a whole because it leaves doors to new passages, to detours and to possibilities of interweaves, of union, of growth and of improvement.

Therefore, it is an architecture that could be defined as being hybrid. Because it grows in the extent that it apprehends from what it is given in its linearity as incorruptible, uncontaminated, as pure theory, as only path, as unique direction, as supreme theory. And incorporates it into a weave that belays it to a random thought, in which its function persists in itself; but still plays one more function with respect to the created totality.

In this respect, it breaks the hackneyed notion of cause and effect; because there are different coincident overlapping causes; uneven diverse causes that generate particular effects around a universal effect, without losing their specific functionality. A diverse whole, interwove of passages, viaducts and connections, which move into precise objectives all its functionality in mathematical terms: an integral that unifies all for the encounter of a primitive totalizing function.

The anatomy of the Rhizome architecture requires such events causing a collision, linking

and finding unions in what it was not meant to be unified and which could have remained isolated if their deep interweaving was not thought of, to happen. The architecture is the result of finding in the unforeseen relationships or links, the reason of a chafe. It is the result of the ability of finding the points of collision or of rubbing that have ties among the most diverse and seemingly distant theories. This unifying thought, requires the rejection of a line of thought over another, of one technique over another, of one way over another.

Educational and learning theorist establish theories that induce to certain practices, to certain techniques; but an anatomist of the thought must find its applicability beyond the ideological quarrels that each theory encompasses. It must solve practically what the theorists propose conceptually.

In the practical-technical solution, an engineer usually does it from a perspective covered by a certain theory leading to surprising technical results. Although always unilateral. An engineer is not interested in the general case because it requires special solutions, particulars to certain problems that have arisen. Then, he must resolve them in the best way. Therefore, he cannot stick to this or that theory, since each one can take him to solve parts of the problem instead of random, broad and integrated solutions.

The engineer considers concrete problems about something, which requires an effective technical solution referring to something concrete, for which a final position is not needed. Thus, limiting himself to a particular theory about knowledge, about learning. Because his problem is not philosophical, is technical.

The solution should cover instead an increasingly wider and complete margin, as a tendency towards an unattainable precise limit, alike the multiplication of the sides of a polygon towards the circumference. The idea of the limit itself. Consequently, the Rhizome architecture works differently because its practical-technical solution cannot be done on a particular determination of this or that theory as engineers do. On the contrary, it observes the established theories as the limits to its action and it tries to link all such theories in a kind of integrative algebra, to find a solution; to find multiple functions for a practical solution because it is not satisfied (and cannot be) of finding partial functions. Therefore, it solves practical problems involving a major number of trends to find wider solutions.

As far as the robotics field is concerned, the Rhizome architecture proposed in this work attempts to involve all theories that could have practical success, for a different solution from the technical point of view. While the theory produces multiple abstract possibilities, the Rhizome architect, must give effective realization to his architecture, at which addresses the possibility of the accomplishment of every theory altogether. The Rhizome architecture assumes this challenge of hybridizing the largest number of theoretical accomplishments, so that the operation is as complete as possible. The Rhizome architect does it by thinking about the possible models that the theorist proposes while observing with a technical eye the practical probability that it offers him: this theory leads to such functioning, such other theory does not; this one comes hitherto, such other goes until there.

The Rhizome architecture unifies into a single project, the possibilities of functioning by involving them and defining practically the scope and limits of one and another. By creating a combinatorial so that if a certain operation does not reach a proposal, another can do it. Therefore, they complement each other and each can be explained by itself and by the others; solving practically what theorists argue on speculation.

Publications

This dissertation has led to the following communications:

Conferences

International

- Dalia Marcela Rojas-Castro, Arnaud Revel, Michel Ménard. "Robotic and Document Analysis Cross-Fertilization: Improving Place Cells Based Robot Navigation". In the proceedings of the 14th International Conference on control, Automation, Robotics and Vision (ICARCV), Phuket, THAILAND, November, 2016.
- Dalia Marcela Rojas-Castro, Arnaud Revel, Michel Ménard. "Artificial Neural Network-Based Control Architecture: a Simultaneous Top-down and Bottom-up Approach to Autonomous Robot Navigation. In the proceedings of the 25th International Conference on Artificial Neural Networks (ICANN), pp 540, Barcelona, SPAIN, September, 2016.
- Dalia Marcela Rojas-Castro, A. Revel, and M. Ménard. "Document image analysis by a mobile robot for autonomous indoor navigation." In the proceedings of the 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 156-160, Nancy, FRANCE, August, 2015.
- Dalia Marcela Rojas-Castro, A. Revel, and M. Ménard. "A Robust Neural Robot Navigation Using a Combination of Deliberative and Reactive Control Architectures." In the proceedings of the 23th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), pp. 445-450, Bruges, BELGIUM, April, 2015.

National

- Dalia Marcela Rojas-Castro, Arnaud Revel, Michel Ménard. "A Hybrid Neural-based Control Architecture for Autonomous Robot Navigation: Synergetic Interface between a priori and embedded information. Congrès National sur la Reconnaissance des Formes et l'intelligence Artificielle/Conférence National en Intelligence Artificielle (RFIA-CNIA), Clermont Ferrand, FRANCE, July, 2016.

Bibliography

- [Abadi 2015] Mohammad Hossein Bamorovat Abadi et Mohammadreza Asghari Oskoei. *Effects of Mirrors in Mobile Robot Navigation Based on Omnidirectional Vision*. In Intelligent Robotics and Applications, pages 37–48. Springer, 2015. (Cited in page 46.)
- [Ahmed 2011] Sheraz Ahmed, Marcus Liwicki, Markus Weber et Andreas Dengel. *Improved automatic analysis of architectural floor plans*. In 2011 International Conference on Document Analysis and Recognition, pages 864–869. IEEE, 2011. (Cité en pages xx, 133, 139 et 141.)
- [Ahn 2006] Sunghwan Ahn, Minyong Choi, Jinwoo Choi et Wan Kyun Chung. *Data association using visual object recognition for EKF-SLAM in home environment*. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 2588–2594. IEEE, 2006. (Cited in page 56.)
- [Alami 1998] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab et Félix Ingrand. *An architecture for autonomy*. The International Journal of Robotics Research, vol. 17, no. 4, pages 315–337, 1998. (Cited in page 67.)
- [Albring Guth 2013] Felipe Albring Guth, Luan Silveira, Marcos Amaral, Silvia Botelho et Paul Drews. *Underwater visual 3D slam using a bio-inspired system*. In Computing and Automation for Offshore Shipbuilding (NAVCOMP), 2013 Symposium on, pages 87–92. IEEE, 2013. (Cited in page 59.)
- [Albus 1991] James S Albus. *Outline for a theory of intelligence*. Systems, Man and Cybernetics, IEEE Transactions on, vol. 21, no. 3, pages 473–509, 1991. (Cited in page 83.)
- [Alonso 2012] Ignacio Parra Alonso, David Fernández Llorca, Miguel Gavilán, Sergio Álvarez Pardo, Miguel Ángel García-Garrido, Ljubo Vlacic et Miguel Ángel Sotelo. *Accurate global localization using visual odometry and digital maps on urban environments*. Intelligent Transportation Systems, IEEE Transactions on, vol. 13, no. 4, pages 1535–1545, 2012. (Cited in page 52.)
- [Alvernhe 2012] Alice Alvernhe, Francesca Sargolini et Bruno Poucet. *Rats build and update topological representations through exploration*. Animal cognition, vol. 15, no. 3, pages 359–368, 2012. (Cited in page 35.)
- [Anati 2012] Roy Anati, Davide Scaramuzza, Konstantinos G Derpanis et Kostas Daniilidis. *Robot localization using soft object detection*. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 4992–4999. IEEE, 2012. (Cited in page 52.)
- [Anderson 2013] Peter Anderson, Yongki Yusmanthia, Bernhard Hengst et Arcot Sowmya. *Robot localisation using natural landmarks*. In RoboCup 2012: Robot Soccer World Cup XVI, pages 118–129. Springer, 2013. (Cited in page 61.)

- [Andrade-Cetto 2002] Juan Andrade-Cetto et Alberto Sanfeliu. *Concurrent map building and localization on indoor dynamic environments*. International Journal of Pattern Recognition and Artificial Intelligence, vol. 16, no. 03, pages 361–374, 2002. (Cited in page 53.)
- [Arbib 1981] Michael A Arbib. *Perceptual structures and distributed motor control*. Comprehensive Physiology, 1981. (Cited in page 83.)
- [Arkin 1987] Ronald C Arkin. *Motor schema based navigation for a mobile robot: An approach to programming by behavior*. In Robotics and Automation. Proceedings. 1987 IEEE International Conference on, volume 4, pages 264–271. IEEE, 1987. (Cité en pages 80 et 83.)
- [Arkin 1989] Ronald C Arkin. *Towards the unification of navigational planning and reactive control*. 1989. (Cité en pages 77 et 80.)
- [Arkin 1990] Ronald C Arkin. *Integrating behavioral, perceptual, and world knowledge in reactive navigation*. Robotics and autonomous systems, vol. 6, no. 1, pages 105–122, 1990. (Cited in page 83.)
- [Arkin 1997] Ronald C Arkin et Tucker Balch. *AuRA: Principles and practice in review*. Journal of Experimental & Theoretical Artificial Intelligence, vol. 9, no. 2-3, pages 175–189, 1997. (Cité en pages 83 et 86.)
- [Arleo 2000] Angelo Arleo et Wulfram Gerstner. *Modeling rodent head-direction cells and place cells for spatial learning in bio-mimetic robotics*. From Animals to Animats, vol. 6, pages 236–245, 2000. (Cited in page 34.)
- [Arleo 2001] A Arleo et F Smeraldi S Hug W Gerstner. *Place Cells and Spatial Navigation based on Vision, Path Integration, and Reinforcement Learning*. In Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference, volume 13, page 89. MIT Press, 2001. (Cited in page 34.)
- [Arleo 2004] Angelo Arleo, Fabrizio Smeraldi et Wulfram Gerstner. *Cognitive navigation based on nonuniform Gabor space sampling, unsupervised growing networks, and reinforcement learning*. IEEE Transactions on Neural Networks, vol. 15, no. 3, pages 639–652, 2004. (Cited in page 34.)
- [Atanasov 2014] Nikolay Atanasov, Menglong Zhu, Kostas Daniilidis et George J Pappas. *Semantic Localization Via the Matrix Permanent*. Proc. Robot.: Sci. Syst, 2014. (Cited in page 53.)
- [Bailey 2006] Tim Bailey et Hugh Durrant-Whyte. *Simultaneous localization and mapping (SLAM): Part II*. IEEE Robotics & Automation Magazine, vol. 13, no. 3, pages 108–117, 2006. (Cited in page 55.)
- [Ballard 1981] Dana H Ballard. *Generalizing the Hough transform to detect arbitrary shapes*. Pattern recognition, vol. 13, no. 2, pages 111–122, 1981. (Cited in page 141.)

- [Banquet 2005] Jean-Paul Banquet, Ph Gaussier, Mathias Quoy, Arnaud Revel et Yves Burnod. *A hierarchy of associations in hippocampo-cortical systems: cognitive maps and navigation strategies*. Neural Computation, vol. 17, no. 6, pages 1339–1384, 2005. (Cit  en pages 35 et 40.)
- [Barrera 2008] Alejandra Barrera et Alfredo Weitzenfeld. *Biologically-inspired robot spatial cognition based on rat neurophysiological studies*. Autonomous Robots, vol. 25, no. 1-2, pages 147–169, 2008. (Cited in page 35.)
- [Bay 2006] Herbert Bay, Tinne Tuytelaars et Luc Van Gool. *Surf: Speeded up robust features*. In European conference on computer vision, pages 404–417. Springer, 2006. (Cited in page 28.)
- [Behzadian 2015] Bahram Behzadian, Pratik Agarwal, Wolfram Burgard et Gian Diego Tipaldi. *Monte Carlo localization in hand-drawn maps*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 4291–4296. IEEE, 2015. (Cited in page 49.)
- [Benet 2002] Gines Benet, Francisco Blanes, Jos  E Sim  et Pascual P rez. *Using infrared sensors for distance measurement in mobile robots*. Robotics and autonomous systems, vol. 40, no. 4, pages 255–266, 2002. (Cited in page 44.)
- [Berg 2010] Alex Berg, Jia Deng et L Fei-Fei. *Large scale visual recognition challenge 2010*, 2010. (Cited in page 28.)
- [Biswas 2012] Joydeep Biswas et Manuela Veloso. *Depth camera based indoor mobile robot localization and navigation*. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 1697–1702. IEEE, 2012. (Cited in page 51.)
- [Bonasso 1995] R Peter Bonasso, David Kortenkamp, David P Miller et Marc Slack. *Experiences with an architecture for intelligent, reactive agents*. In Intelligent Agents II Agent Theories, Architectures, and Languages, pages 187–202. Springer, 1995. (Cited in page 86.)
- [Boniardi 2015] Federico Boniardi, Bahram Behzadian, Wolfram Burgard et Gian Diego Tipaldi. *Robot navigation in hand-drawn sketched maps*. In Mobile Robots (ECMR), 2015 European Conference on, pages 1–6. IEEE, 2015. (Cited in page 49.)
- [Bonin-Font 2008] Francisco Bonin-Font, Alberto Ortiz et Gabriel Oliver. *Visual navigation for mobile robots: A survey*. Journal of intelligent and robotic systems, vol. 53, no. 3, pages 263–296, 2008. (Cited in page 22.)
- [Braitenberg 1986] Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT press, 1986. (Cit  en pages xviii, 73, 75 et 77.)
- [Brooks 1986] Rodney A Brooks. *A robust layered control system for a mobile robot*. Robotics and Automation, IEEE Journal of, vol. 2, no. 1, pages 14–23, 1986. (Cit  en pages xviii, 10, 11, 67, 71, 72, 74, 75, 80, 81, 82 et 83.)

- [Brooks 1990] Rodney A Brooks. *Elephants don't play chess*. Robotics and autonomous systems, vol. 6, no. 1, pages 3–15, 1990. (Cité en pages 82 et 83.)
- [Brooks 1991] Rodney A Brooks. *Intelligence without representation*. Artificial intelligence, vol. 47, no. 1, pages 139–159, 1991. (Cited in page 82.)
- [Brown 1995] Michael A Brown et Patricia E Sharp. *Simulation of spatial learning in the Morris water maze by a neural network model of the hippocampal formation and nucleus accumbens*. Hippocampus, vol. 5, no. 3, pages 171–188, 1995. (Cited in page 33.)
- [Bugmann 1995] Guido Bugmann, John G Taylor et M Denham. *Route finding by neural nets*. Neural networks, pages 217–230, 1995. (Cité en pages xx et 143.)
- [Burgess 1994] Neil Burgess, Michael Recce et John O'Keefe. *A model of hippocampal function*. Neural networks, vol. 7, no. 6-7, pages 1065–1081, 1994. (Cited in page 33.)
- [Burguera 2014] Antoni Burguera, Francisco Bonin-Font et Gabriel Oliver. *Towards robust image registration for underwater visual SLAM*. In Computer Vision Theory and Applications (VISAPP), 2014 International Conference on, volume 3, pages 539–544. IEEE, 2014. (Cited in page 56.)
- [Burnod 1990] Yves Burnod. *An adaptive neural network: the cerebral cortex*. Masson editeur, 1990. (Cité en pages 36 et 83.)
- [Cabezas 2012] Ivan Cabezas, Maria Trujillo et Margaret Florian. *An Evaluation Methodology for Stereo Correspondence Algorithms*. In VISAPP (2), pages 154–163, 2012. (Cited in page 46.)
- [Calonder 2010] Michael Calonder, Vincent Lepetit, Christoph Strecha et Pascal Fua. *Brief: Binary robust independent elementary features*. In European conference on computer vision, pages 778–792. Springer, 2010. (Cited in page 28.)
- [Caluwaerts 2012] Ken Caluwaerts, Mariacarla Staffa, Steve N'Guyen, Christophe Grand, Laurent Dollé, Antoine Favre-Félix, Benoît Girard et Mehdi Khamassi. *A biologically inspired meta-control navigation system for the psikharpax rat robot*. Bioinspiration & biomimetics, vol. 7, no. 2, page 025009, 2012. (Cited in page 35.)
- [Camus 1997] Ted Camus. *Real-time quantized optical flow*. Real-Time Imaging, vol. 3, no. 2, pages 71–86, 1997. (Cited in page 60.)
- [Canny 1986] John Canny. *A computational approach to edge detection*. IEEE Transactions on pattern analysis and machine intelligence, no. 6, pages 679–698, 1986. (Cited in page 26.)
- [Carpenter 1987] Gail A Carpenter et Stephen Grossberg. *A massively parallel architecture for a self-organizing neural pattern recognition machine*. Computer vision, graphics, and image processing, vol. 37, no. 1, pages 54–115, 1987. (Cité en pages 11, 12, 13, 29, 31 et 83.)

- [Cartwright 1983] BA Cartwright et Thomas S Collett. *Landmark learning in bees*. Journal of comparative physiology, vol. 151, no. 4, pages 521–543, 1983. (Cité en pages 33 et 62.)
- [Castellanos 2001] José A Castellanos, José Neira et Juan D Tardós. *Multisensor fusion for simultaneous localization and map building*. Robotics and Automation, IEEE Transactions on, vol. 17, no. 6, pages 908–914, 2001. (Cited in page 45.)
- [Chao 2014] Haiyang Chao, Yu Gu et Marcello Napolitano. *A survey of optical flow techniques for robotics navigation applications*. Journal of Intelligent & Robotic Systems, vol. 73, no. 1-4, pages 361–372, 2014. (Cited in page 60.)
- [Chatila 1985] Raja Chatila et Jean-Paul Laumond. *Position referencing and consistent world modeling for mobile robots*. In Robotics and Automation. Proceedings. 1985 IEEE International Conference on, volume 2, pages 138–145. IEEE, 1985. (Cité en pages 53 et 72.)
- [Choset 2005] Howie M Choset. Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005. (Cité en pages xviii et 48.)
- [Chum 2007] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard et Andrew Zisserman. *Total recall: Automatic query expansion with a generative feature model for object retrieval*. In 2007 IEEE 11th International Conference on Computer Vision, pages 1–8. IEEE, 2007. (Cited in page 28.)
- [Civera 2010] Javier Civera, Oscar G Grasa, Andrew J Davison et JMM Montiel. *1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry*. Journal of Field Robotics, vol. 27, no. 5, pages 609–631, 2010. (Cited in page 56.)
- [Clancey 1991] William J Clancey. *The frame of reference problem in the design of intelligent machines*. Architectures for intelligence, pages 357–423, 1991. (Cited in page 69.)
- [Clark 2002] Paul Clark et Majid Mirmehdi. *Recognising text in real scenes*. International Journal on Document Analysis and Recognition, vol. 4, no. 4, pages 243–257, 2002. (Cited in page 140.)
- [Clark 2003] Paul Clark et Majid Mirmehdi. *Rectifying perspective views of text in 3D scenes using vanishing points*. Pattern Recognition, vol. 36, no. 11, pages 2673–2686, 2003. (Cited in page 138.)
- [Clemente 2007] Laura A Clemente, Andrew J Davison, Ian D Reid, José Neira et Juan D Tardós. *Mapping Large Loops with a Single Hand-Held Camera*. In Robotics: Science and Systems, volume 2, page 2, 2007. (Cited in page 56.)
- [Connell 1992] Jonathan H Connell. *SSS: A hybrid architecture applied to robot navigation*. In Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, pages 2719–2724. IEEE, 1992. (Cited in page 77.)

- [Connolly 1990] Christopher I Connolly, John B Burns et R Weiss. *Path planning using Laplace's equation*. In Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, pages 2102–2106. IEEE, 1990. (Cited in page 143.)
- [Csurka 2004] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski et Cédric Bray. *Visual categorization with bags of keypoints*. In Workshop on statistical learning in computer vision, ECCV, volume 1, pages 1–2. Prague, 2004. (Cited in page 28.)
- [Cummins 2008] Mark Cummins et Paul Newman. *FAB-MAP: Probabilistic localization and mapping in the space of appearance*. The International Journal of Robotics Research, vol. 27, no. 6, pages 647–665, 2008. (Cité en pages 32, 57 et 59.)
- [Cuperlier 2007] Nicolas Cuperlier, Mathias Quoy et Philippe Gaussier. *Neurobiologically inspired mobile robot navigation and planning*. Frontiers in neurorobotics, vol. 1, page 3, 2007. (Cité en pages xvii et 37.)
- [Dance 2001] Christopher R Dance. *Perspective estimation for document images*. In Electronic Imaging 2002, pages 244–254. International Society for Optics and Photonics, 2001. (Cited in page 138.)
- [Davies 2008] Trevor Davies et Amor Jnifene. *Path planning and trajectory control of collaborative mobile robots using hybrid control architecture*. Journal of Systemics, Cybernetics and Informatics, vol. 6, no. 4, pages 42–48, 2008. (Cited in page 78.)
- [Davison 2003] Andrew J Davison. *Real-time simultaneous localisation and mapping with a single camera*. In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pages 1403–1410. IEEE, 2003. (Cited in page 56.)
- [Davison 2007] Andrew J Davison, Ian D Reid, Nicholas D Molton et Olivier Stasse. *MonoSLAM: Real-time single camera SLAM*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, no. 6, pages 1052–1067, 2007. (Cited in page 56.)
- [de la Puente 2014] Paloma de la Puente et Diego Rodríguez-Losada. *Feature based graph-SLAM in structured environments*. Autonomous Robots, vol. 37, no. 3, pages 243–260, 2014. (Cited in page 48.)
- [Deriche 1987] Rachid Deriche. *Using Canny's criteria to derive a recursively implemented optimal edge detector*. International journal of computer vision, vol. 1, no. 2, pages 167–187, 1987. (Cited in page 26.)
- [DeSouza 2002] Guilherme N DeSouza et Avinash C Kak. *Vision for mobile robot navigation: A survey*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 2, pages 237–267, 2002. (Cité en pages 21 et 51.)
- [Doeller 2010] Christian F Doeller, Caswell Barry et Neil Burgess. *Evidence for grid cells in a human memory network*. Nature, vol. 463, no. 7281, pages 657–661, 2010. (Cited in page 35.)

- [Dollé 2010] Laurent Dollé, Denis Sheynikhovich, Benoît Girard, Ricardo Chavarriaga et Agnès Guillot. *Path planning versus cue responding: a bio-inspired model of switching between navigation strategies*. Biological cybernetics, vol. 103, no. 4, pages 299–317, 2010. (Cited in page 35.)
- [Donahue 2014] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng et Trevor Darrell. *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition*. In ICML, pages 647–655, 2014. (Cited in page 33.)
- [Dosch 2000] Philippe Dosch, Karl Tombre, Christian Ah-Soon et Gérald Masini. *A complete system for the analysis of architectural drawings*. International Journal on Document Analysis and Recognition, vol. 3, no. 2, pages 102–116, 2000. (Cited in page 141.)
- [Durrant-Whyte 1988] Hugh F Durrant-Whyte. *Uncertain geometry in robotics*. Robotics and Automation, IEEE Journal of, vol. 4, no. 1, pages 23–31, 1988. (Cited in page 53.)
- [Durrant-Whyte 1996] Hugh Durrant-Whyte, David Rye et Eduardo Nebot. *Localization of autonomous guided vehicles*. In Robotics Research, pages 613–625. Springer, 1996. (Cited in page 53.)
- [Durrant-Whyte 2006] Hugh Durrant-Whyte et Tim Bailey. *Simultaneous localization and mapping: part I*. Robotics & Automation Magazine, IEEE, vol. 13, no. 2, pages 99–110, 2006. (Cited in page 55.)
- [Eade 2006] Ethan Eade et Tom Drummond. *Scalable monocular SLAM*. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 469–476. IEEE, 2006. (Cited in page 57.)
- [Eade 2008] Ethan Eade et Tom Drummond. *Unified Loop Closing and Recovery for Real Time Monocular SLAM*. In BMVC, volume 13, page 136, 2008. (Cited in page 32.)
- [Edelman 1987] Gerald M Edelman. Neural darwinism: The theory of neuronal group selection. Basic Books, 1987. (Cité en pages 10, 11, 12 et 83.)
- [Einhorn 2015] Erik Einhorn et Horst-Michael Gross. *Generic NDT mapping in dynamic environments and its application for lifelong SLAM*. Robotics and Autonomous Systems, vol. 69, pages 28–39, 2015. (Cited in page 53.)
- [Ekstrom 2003] Arne D Ekstrom, Michael J Kahana, Jeremy B Caplan, Tony A Fields, Eve A Isham, Ehren L Newman et Itzhak Fried. *Cellular networks underlying human spatial navigation*. Nature, vol. 425, no. 6954, pages 184–188, 2003. (Cited in page 34.)
- [Elfes 2013] Alberto Elfes. *Occupancy grids: A stochastic spatial representation for active robot perception*. arXiv preprint arXiv:1304.1098, 2013. (Cited in page 48.)
- [Engel 2014] Jakob Engel, Jürgen Sturm et Daniel Cremers. *Scale-aware navigation of a low-cost quadcopter with a monocular camera*. Robotics and Autonomous Systems, vol. 62, no. 11, pages 1646–1656, 2014. (Cited in page 45.)

- [Engel 2015] Jakob Engel, Jorg Stuckler et Daniel Cremers. *Large-scale direct slam with stereo cameras*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 1935–1942. IEEE, 2015. (Cited in page 46.)
- [Erinc 2014] Gorkem Erinc et Stefano Carpin. *Anytime merging of appearance-based maps*. Autonomous Robots, vol. 36, no. 3, pages 241–256, 2014. (Cité en pages 48 et 49.)
- [Eudes 2010] Alexandre Eudes, Maxime Lhuillier, Sylvie Naudet-Collette et Michel Dhome. *Fast odometry integration in local bundle adjustment-based visual slam*. In Pattern Recognition (ICPR), 2010 20th International Conference on, pages 290–293. IEEE, 2010. (Cited in page 58.)
- [Everingham 2008] M Everingham, L Van Gool, CKI Williams, J Winn et A Zisserman. *The pascal visual object classes challenge 2007 (voc 2007) results (2007)*, 2008. (Cited in page 28.)
- [Fallon 2013] Maurice Fallon, Hordur Johannsson, Michael Kaess et John J Leonard. *The MIT stata center dataset*. The International Journal of Robotics Research, vol. 32, no. 14, pages 1695–1699, 2013. (Cited in page 49.)
- [Fazl-Ersi 2012] Ehsan Fazl-Ersi et John K Tsotsos. *Histogram of oriented uniform patterns for robust place recognition and categorization*. The International Journal of Robotics Research, page 0278364911434936, 2012. (Cited in page 33.)
- [Fernandes 2012] Cláudio dos S Fernandes, Mario FM Campos et Luiz Chaimowicz. *A low-cost localization system based on Artificial Landmarks*. In Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian, pages 109–114. IEEE, 2012. (Cited in page 62.)
- [Filliat 2002] David Filliat, Jean-Arcady Meyer et al. *Global localization and topological map-learning for robot navigation*. From animals to animats, vol. 7, pages 131–140, 2002. (Cited in page 33.)
- [Floros 2013] Georgios Floros, Benito van der Zander et Bastian Leibe. *Openstreetslam: Global vehicle localization using openstreetmaps*. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 1054–1059. IEEE, 2013. (Cited in page 49.)
- [Fox 1999] Dieter Fox, Wolfram Burgard et Sebastian Thrun. *Markov localization for mobile robots in dynamic environments*. Journal of Artificial Intelligence Research, pages 391–427, 1999. (Cited in page 52.)
- [Frese 2010] Udo Frese, René Wagner et Thomas Röfer. *A SLAM Overview from a User's Perspective*. KI-Künstliche Intelligenz, vol. 24, no. 3, pages 191–198, 2010. (Cited in page 53.)
- [Frintrop 2008] Simone Frintrop et Patric Jensfelt. *Attentional landmarks and active gaze control for visual SLAM*. Robotics, IEEE Transactions on, vol. 24, no. 5, pages 1054–1065, 2008. (Cited in page 49.)

- [Fuentes-Pacheco 2015] Jorge Fuentes-Pacheco, José Ruiz-Ascencio et Juan Manuel Rendón-Mancha. *Visual simultaneous localization and mapping: a survey*. Artificial Intelligence Review, vol. 43, no. 1, pages 55–81, 2015. (Cited in page 49.)
- [Fyhn 2007] Marianne Fyhn, Torkel Hafting, Alessandro Treves, May-Britt Moser et Edvard I Moser. *Hippocampal remapping and grid realignment in entorhinal cortex*. Nature, vol. 446, no. 7132, pages 190–194, 2007. (Cited in page 35.)
- [Gallistel 1993] C Randy Gallistel. Organization of learning (learning, development, and conceptual change). MIT Press, 1993. (Cited in page 33.)
- [Garcia-Fidalgo 2015] Emilio Garcia-Fidalgo et Alberto Ortiz. *Vision-based topological mapping and localization methods: A survey*. Robotics and Autonomous Systems, vol. 64, pages 1–20, 2015. (Cited in page 48.)
- [Gaussier 1995] Philippe Gaussier et Stéphane Zrehen. *Perac: A neural architecture to control artificial animals*. Robotics and Autonomous Systems, vol. 16, no. 2, pages 291–320, 1995. (Cité en pages xvii, xviii, 11, 14, 16, 37, 82, 83, 85 et 134.)
- [Gaussier 1997] Philippe Gaussier, Cédric Joulain, Stéphane Zrehen, J-P Banquet et Arnaud Revel. *Visual navigation in an open environment without map*. In Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on, volume 2, pages 545–550. IEEE, 1997. (Cité en pages xvii, 35, 36 et 62.)
- [Gaussier 2000] Philippe Gaussier, Cédric Joulain, Jean-Paul Banquet, Sacha Leprêtre et Arnaud Revel. *The visual homing problem: an example of robotics/biology cross fertilization*. Robotics and autonomous systems, vol. 30, no. 1, pages 155–180, 2000. (Cité en pages xvii, 34, 35, 37 et 38.)
- [Gaussier 2002] Philippe Gaussier, Arnaud Revel, Jean-Paul Banquet et Vincent Babeau. *From view cells and place cells to cognitive map learning: processing stages of the hippocampal system*. Biological cybernetics, vol. 86, no. 1, pages 15–28, 2002. (Cité en pages 11, 34 et 35.)
- [Gaussier 2007] Ph Gaussier, JP Banquet, F Sargolini, C Giovannangeli, E Save et B Poucet. *A model of grid cells involving extra hippocampal path integration, and the hippocampal loop*. Journal of integrative neuroscience, vol. 6, no. 03, pages 447–476, 2007. (Cited in page 35.)
- [Geng 2009] Xin Geng et Kate Smith-Miles. *Facial age estimation by multilinear subspace analysis*. In 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 865–868. IEEE, 2009. (Cited in page 29.)
- [Gibson 1950] James J Gibson. The perception of the visual world. Houghton Mifflin, 1950. (Cited in page 60.)
- [Gilbert 1983] Charles D Gilbert. *Microcircuitry of the visual cortex*. Annual review of neuroscience, vol. 6, no. 1, pages 217–247, 1983. (Cited in page 36.)

- [Giovannangeli 2006a] C Giovannangeli, Ph Gaussier et JP Banquet. *Robustness of visual place cells in dynamic indoor and outdoor environment*. International Journal of Advanced Robotic Systems, vol. 3, no. 2, pages 115–124, 2006. (Cité en pages xvii, 34, 35, 37, 40 et 41.)
- [Giovannangeli 2006b] Christophe Giovannangeli, Philippe Gaussier et Gaël Désilles. *Robust mapless outdoor vision-based navigation*. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 3293–3300. IEEE, 2006. (Cité en pages 35 et 62.)
- [Giovannangeli 2008] Christophe Giovannangeli et Philippe Gaussier. *Autonomous vision-based navigation: Goal-oriented action planning by transient states prediction, cognitive map building, and sensory-motor learning*. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 676–683. IEEE, 2008. (Cited in page 35.)
- [Glover 2010] Arren J Glover, William P Maddern, Michael J Milford et Gordon F Wyeth. *FAB-MAP+ RatSLAM: Appearance-based SLAM for multiple times of day*. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 3507–3512. IEEE, 2010. (Cited in page 59.)
- [Goodale 1992] Melvyn A Goodale et A David Milner. *Separate visual pathways for perception and action*. Trends in neurosciences, vol. 15, no. 1, pages 20–25, 1992. (Cited in page 36.)
- [Grauman 2011] Kristen Grauman et Bastian Leibe. *Visual object recognition*. Synthesis lectures on artificial intelligence and machine learning, vol. 5, no. 2, pages 1–181, 2011. (Cited in page 268.)
- [Guazzelli 1998] Alex Guazzelli, Mihail Bota, Fernando J Corbacho et Michael A Arbib. *Affordances. Motivations, and the World Graph Theory*. Adaptive Behavior, vol. 6, no. 3-4, pages 435–471, 1998. (Cited in page 33.)
- [Guzel 2012] Mehmet Serdar Guzel et Robert Bicker. *A behaviour-based architecture for mapless navigation using vision*. International Journal of Advanced Robotic Systems, vol. 9, 2012. (Cited in page 62.)
- [Güzel 2013] Mehmet Serdar Güzel. *Autonomous vehicle navigation using vision and mapless strategies: a survey*. Advances in Mechanical Engineering, vol. 5, page 234747, 2013. (Cité en pages xvii et 22.)
- [Hafting 2005] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser et Edvard I Moser. *Microstructure of a spatial map in the entorhinal cortex*. Nature, vol. 436, no. 7052, pages 801–806, 2005. (Cited in page 35.)
- [Harms 2015] Hannes Harms, Eike Rehder, Tobias Schwarze et Martin Lauer. *Detection of ascending stairs using stereo vision*. In Intelligent Robots and Systems (IROS),

- 2015 IEEE/RSJ International Conference on, pages 2496–2502. IEEE, 2015. (Cited in page 46.)
- [Harnad 1990] Stevan Harnad. *The symbol grounding problem*. Physica D: Nonlinear Phenomena, vol. 42, no. 1-3, pages 335–346, 1990. (Cited in page 9.)
- [Hebb 2005] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005. (Cited in page 10.)
- [Hecht-Nielsen 1987] Robert Hecht-Nielsen. *Counterpropagation networks*. Applied optics, vol. 26, no. 23, pages 4979–4984, 1987. (Cited in page 83.)
- [Hentschel 2010] Matthias Hentschel et Bernardo Wagner. *Autonomous robot navigation based on openstreetmap geodata*. In Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on, pages 1645–1650. IEEE, 2010. (Cited in page 49.)
- [Ho 2007] Kin Leong Ho et Paul Newman. *Detecting loop closure with scene sequences*. International Journal of Computer Vision, vol. 74, no. 3, pages 261–286, 2007. (Cited in page 32.)
- [Honegger 2013] Dominik Honegger, Lorenz Meier, Petri Tanskanen et Marc Pollefeys. *An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications*. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 1736–1741. IEEE, 2013. (Cited in page 60.)
- [Horn 1981] Berthold K Horn et Brian G Schunck. *Determining optical flow*. In 1981 Technical symposium east, pages 319–331. International Society for Optics and Photonics, 1981. (Cited in page 60.)
- [Huo 1997] Qiang Huo et Chin-Hui Lee. *On-line adaptive learning of the continuous density hidden Markov model based on approximate recursive Bayes estimate*. IEEE transactions on speech and audio processing, vol. 5, no. 2, pages 161–172, 1997. (Cited in page 31.)
- [Indelman 2015] Vadim Indelman, Richard Roberts et Frank Dellaert. *Incremental light bundle adjustment for structure from motion and robotics*. Robotics and Autonomous Systems, vol. 70, pages 63–82, 2015. (Cited in page 58.)
- [Irie 2015] Kiyoshi Irie, Masashi Sugiyama et Masahiro Tomono. *A dependence maximization approach towards street map-based localization*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 3721–3728. IEEE, 2015. (Cited in page 52.)
- [Ito 2014] Satoshi Ito, Felix Endres, Markus Kuderer, Gian Diego Tipaldi, Cyrill Stachniss et Wolfram Burgard. *W-rgb-d: floor-plan-based indoor global localization using a depth camera and wifi*. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 417–422. IEEE, 2014. (Cité en pages 49 et 52.)

- [Jacobs 2013] Joshua Jacobs, Christoph T Weidemann, Jonathan F Miller, Alec Solway, John F Burke, Xue-Xin Wei, Nanthia Suthana, Michael R Sperling, Ashwini D Sharan, Itzhak Fried *et al.* *Direct recordings of grid-like neuronal activity in human spatial navigation.* Nature neuroscience, vol. 16, no. 9, pages 1188–1190, 2013. (Cited in page 35.)
- [Jauffret 2012] Adrien Jauffret, Nicolas Cuperlier, Philippe Gaussier et Philippe Tarroux. *Multimodal integration of visual place cells and grid cells for navigation tasks of a real robot.* In International Conference on Simulation of Adaptive Behavior, pages 136–145. Springer, 2012. (Cited in page 35.)
- [Jensfelt 2001] Patric Jensfelt et Steen Kristensen. *Active global localization for a mobile robot using multiple hypothesis tracking.* Robotics and Automation, IEEE Transactions on, vol. 17, no. 5, pages 748–760, 2001. (Cited in page 52.)
- [Jessup 2014] J Jessup, Sidney Nascimento Givigi et Alain Beaulieu. *Robust and efficient multi-robot 3d mapping with octree based occupancy grids.* In 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 3996–4001. IEEE, 2014. (Cited in page 48.)
- [Jiang 2013] Haibo Jiang, Yilong Xiao, Yunwei Zhang, Xiaojing Wang et Haijiang Tai. *Curve path detection of unstructured roads for the outdoor robot navigation.* Mathematical and Computer Modelling, vol. 58, no. 3, pages 536–544, 2013. (Cited in page 21.)
- [Johannsson 2013] Hordur Johannsson, Michael Kaess, Maurice Fallon et John J Leonard. *Temporally scalable visual SLAM using a reduced pose graph.* In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 54–61. IEEE, 2013. (Cited in page 55.)
- [Johnson 2012] Collin Johnson et Benjamin Kuipers. *Efficient search for correct and useful topological maps.* In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5277–5282. IEEE, 2012. (Cited in page 48.)
- [Jones 2011] Eagle S Jones et Stefano Soatto. *Visual-inertial navigation, mapping and localization: A scalable real-time causal approach.* The International Journal of Robotics Research, vol. 30, no. 4, pages 407–430, 2011. (Cited in page 56.)
- [Joubert 2015] Daniek Joubert, Willie Brink et Ben Herbst. *Pose uncertainty in occupancy grids through Monte Carlo integration.* Journal of Intelligent & Robotic Systems, vol. 77, no. 1, pages 5–16, 2015. (Cited in page 48.)
- [Judd 1998] SPD Judd et TS Collett. *Multiple stored views and landmark guidance in ants.* Nature, vol. 392, no. 6677, pages 710–714, 1998. (Cited in page 33.)
- [Jung 1993] MW Jung et BL McNaughton. *Spatial selectivity of unit activity in the hippocampal granular layer.* Hippocampus, vol. 3, no. 2, pages 165–182, 1993. (Cited in page 34.)

- [Kawamura 2002] Kazuhiko Kawamura, AB Koku, D Mitchell Wilkes, Richard Alan Peters et A Sekmen. *Toward egocentric navigation*. International Journal of Robotics and Automation, vol. 17, no. 4, pages 135–145, 2002. (Cited in page 49.)
- [Kerl 2013] Christian Kerl, Jurgen Sturm et Daniel Cremers. *Dense visual slam for rgb-d cameras*. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 2100–2106. IEEE, 2013. (Cited in page 55.)
- [Kidono 2002] Kiyosumi Kidono, Jun Miura et Yoshiaki Shirai. *Autonomous visual navigation of a mobile robot using a human-guided experience*. Robotics and Autonomous Systems, vol. 40, no. 2, pages 121–130, 2002. (Cited in page 54.)
- [Killian 2012] Nathaniel J Killian, Michael J Jutras et Elizabeth A Buffalo. *A map of visual space in the primate entorhinal cortex*. Nature, vol. 491, no. 7426, pages 761–764, 2012. (Cited in page 35.)
- [Knopp 2010] Jan Knopp, Josef Sivic et Tomas Pajdla. *Avoiding confusing features in place recognition*. In European Conference on Computer Vision, pages 748–761. Springer, 2010. (Cited in page 32.)
- [Kohonen 1990] Teuvo Kohonen. *The self-organizing map*. Proceedings of the IEEE, vol. 78, no. 9, pages 1464–1480, 1990. (Cited in page 12.)
- [Konolige 1997] Kurt Konolige, Karen Myers, Enrique Ruspini et Alessandro Saffiotti. *The Saphira architecture: A design for autonomy*. Journal of experimental & theoretical artificial intelligence, vol. 9, no. 2-3, pages 215–235, 1997. (Cited in page 78.)
- [Kosaka 1992] Akio Kosaka et Avinash C Kak. *Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties*. CVGIP: Image understanding, vol. 56, no. 3, pages 271–329, 1992. (Cited in page 51.)
- [Krizhevsky 2012] Alex Krizhevsky, Ilya Sutskever et Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. In Advances in neural information processing systems, pages 1097–1105, 2012. (Cited in page 33.)
- [Latombe 1991] Jean-Claude Latombe, Anthony Lazanas et Shashank Shekhar. *Robot motion planning with uncertainty in control and sensing*. Artificial Intelligence, vol. 52, no. 1, pages 1–47, 1991. (Cited in page 72.)
- [Lemaire 2007] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung et Simon Lacroix. *Vision-based slam: Stereo and monocular approaches*. International Journal of Computer Vision, vol. 74, no. 3, pages 343–364, 2007. (Cited in page 55.)
- [Leonard 1991] John J Leonard et Hugh F Durrant-Whyte. *Simultaneous map building and localization for an autonomous mobile robot*. In Intelligent Robots and Systems' 91. Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on, pages 1442–1447. Ieee, 1991. (Cited in page 53.)

- [Li 2015] Xiuzhi Li et Huan Qiu. *An effective laser-based approach to build topological map of unknown environment*. In 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 200–205. IEEE, 2015. (Cited in page 48.)
- [Lindeberg 2012] Tony Lindeberg. *Scale invariant feature transform*. Scholarpedia, vol. 7, no. 5, page 10491, 2012. (Cited in page 27.)
- [Lindström 2000] Mattias Lindström, Anders Orebäck et Henrik I Christensen. *Berra: A research architecture for service robots*. In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, volume 4, pages 3278–3283. IEEE, 2000. (Cited in page 77.)
- [Lodhi 2002] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini et Chris Watkins. *Text classification using string kernels*. Journal of Machine Learning Research, vol. 2, no. Feb, pages 419–444, 2002. (Cited in page 28.)
- [Low 2002] Kian Hsiang Low, Wee Kheng Leow et Marcelo H Ang Jr. *A hybrid mobile robot architecture with integrated planning and control*. In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, pages 219–226. ACM, 2002. (Cited in page 77.)
- [Lowe 2004] David G Lowe. *Distinctive image features from scale-invariant keypoints*. International journal of computer vision, vol. 60, no. 2, pages 91–110, 2004. (Cité en pages 13, 26, 28, 63, 167 et 263.)
- [Lu 2005] Shijian Lu, Ben M Chen et Chi Chung Ko. *Perspective rectification of document images using fuzzy set and morphological operations*. Image and Vision Computing, vol. 23, no. 5, pages 541–553, 2005. (Cited in page 138.)
- [Lucas 1981] Bruce D Lucas, Takeo Kanade et al. *An iterative image registration technique with an application to stereo vision*. In IJCAI, volume 81, pages 674–679, 1981. (Cited in page 60.)
- [Macé 2010] Sébastien Macé, Hervé Locteau, Ernest Valveny et Salvatore Tabbone. *A system to detect rooms in architectural floor plan images*. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, pages 167–174. ACM, 2010. (Cited in page 141.)
- [Maddern 2012] Will Maddern, Michael Milford et Gordon Wyeth. *CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory*. The International Journal of Robotics Research, vol. 31, no. 4, pages 429–451, 2012. (Cited in page 59.)
- [Maes 1989] Pattie Maes. The dynamics of action selection. Artificial Intelligence Laboratory, Vrije Universiteit Brussel, 1989. (Cited in page 80.)
- [Maes 1990] Pattie Maes. *Situated agents can have goals*. Robotics and autonomous systems, vol. 6, no. 1, pages 49–70, 1990. (Cited in page 80.)

- [Maohai 2013] Li Maohai, Wang Han, Sun Lining et Cai Zesu. *Robust omnidirectional mobile robot topological navigation system using omnidirectional vision*. Engineering applications of artificial intelligence, vol. 26, no. 8, pages 1942–1952, 2013. (Cited in page 46.)
- [Mataric 1991] Maja J Mataric. *Navigating with a rat brain: a neurobiologically inspired model*. In From Animals to Animats; Proceedings of the First International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge, Mass, 1991. (Cited in page 81.)
- [Mataric 1992] Maja J Mataric. *Behavior-based control: Main properties and implications*. In Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems, pages 46–54. Citeseer, 1992. (Cited in page 67.)
- [McClelland 1986] James L McClelland, David E Rumelhart et Geoffrey E Hinton. *The appeal of parallel distributed processing*. MIT Press, Cambridge MA, pages 3–44, 1986. (Cited in page 10.)
- [McGann 2008] Conor McGann, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn et Rob McEwen. *A deliberative architecture for AUV control*. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 1049–1054. IEEE, 2008. (Cited in page 73.)
- [McNaughton 2006] Bruce L McNaughton, Francesco P Battaglia, Ole Jensen, Edvard I Moser et May-Britt Moser. *Path integration and the neural basis of the 'cognitive map'*. Nature Reviews Neuroscience, vol. 7, no. 8, pages 663–678, 2006. (Cited in page 35.)
- [Meng 1993a] Min Meng et Avinash C Kak. *Mobile robot navigation using neural networks and nonmetrical environmental models*. Control Systems, IEEE, vol. 13, no. 5, pages 30–39, 1993. (Cited in page 51.)
- [Meng 1993b] Min Meng et Avinash C Kak. *NEURO-NAV: a neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment*. In Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on, pages 750–757. IEEE, 1993. (Cited in page 51.)
- [Meyer-Delius 2012] Daniel Meyer-Delius, Maximilian Beinhofer et Wolfram Burgard. *Occupancy Grid Models for Robot Mapping in Changing Environments*. In AAI, 2012. (Cité en pages 48 et 54.)
- [Michaud 1997] François Michaud. *Selecting behaviors using fuzzy logic*. In Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on, volume 1, pages 585–592. IEEE, 1997. (Cited in page 80.)
- [Mikolajczyk 2004] Krystian Mikolajczyk et Cordelia Schmid. *Scale & affine invariant interest point detectors*. International journal of computer vision, vol. 60, no. 1, pages 63–86, 2004. (Cited in page 27.)

- [Milford 2004] Michael J Milford, Gordon F Wyeth et DF Rasser. *RatSLAM: a hippocampal model for simultaneous localization and mapping*. In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, volume 1, pages 403–408. IEEE, 2004. (Cité en pages 35 et 58.)
- [Milford 2007] Michael Milford et Gordon Wyeth. *Spatial mapping and map exploitation: a bio-inspired engineering perspective*. In International Conference on Spatial Information Theory, pages 203–221. Springer, 2007. (Cited in page 35.)
- [Milford 2008a] Michael J Milford et Gordon F Wyeth. *Mapping a suburb with a single camera using a biologically inspired SLAM system*. Robotics, IEEE Transactions on, vol. 24, no. 5, pages 1038–1053, 2008. (Cited in page 58.)
- [Milford 2008b] Michael J Milford et Gordon F Wyeth. *Single camera vision-only SLAM on a suburban road network*. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 3684–3689. IEEE, 2008. (Cited in page 56.)
- [Milford 2010] Michael Milford et Gordon Wyeth. *Persistent navigation and mapping using a biologically inspired SLAM system*. The International Journal of Robotics Research, vol. 29, no. 9, pages 1131–1153, 2010. (Cited in page 35.)
- [Mishkin 1982] Mortimer Mishkin et Leslie G Ungerleider. *Contribution of striate inputs to the visuospatial functions of parieto-preoccipital cortex in monkeys*. Behavioural brain research, vol. 6, no. 1, pages 57–77, 1982. (Cited in page 36.)
- [Montemerlo 2002] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit et al. *FastSLAM: A factored solution to the simultaneous localization and mapping problem*. In Aaai/iaai, pages 593–598, 2002. (Cited in page 57.)
- [Montiel 2006] JMM Montiel, Javier Civera et Andrew J Davison. *Unified inverse depth parametrization for monocular SLAM*. analysis, vol. 9, page 1, 2006. (Cited in page 56.)
- [Moravec 1985] Hans P Moravec et Alberto Elfes. *High resolution maps from wide angle sonar*. In Robotics and Automation. Proceedings. 1985 IEEE International Conference on, volume 2, pages 116–121. IEEE, 1985. (Cité en pages 27 et 54.)
- [Morris 1981] Richard GM Morris. *Spatial localization does not require the presence of local cues*. Learning and motivation, vol. 12, no. 2, pages 239–260, 1981. (Cited in page 35.)
- [Morris 1984] Richard Morris. *Developments of a water-maze procedure for studying spatial learning in the rat*. Journal of neuroscience methods, vol. 11, no. 1, pages 47–60, 1984. (Cited in page 35.)
- [Moser 2008] Edvard I Moser, Emilio Kropff et May-Britt Moser. *Place cells, grid cells, and the brain's spatial representation system*. Neuroscience, vol. 31, no. 1, page 69, 2008. (Cited in page 34.)

- [Mouragnon 2006] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser et Patrick Sayd. *Monocular vision based SLAM for mobile robots*. In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, volume 3, pages 1027–1031. IEEE, 2006. (Cited in page 58.)
- [Moya-Albor 2016] Ernesto Moya-Albor, Jorge Brieva et Hiram Eredín Ponce Espinosa. *Mobile Robot with Movement Detection Controlled by a Real-Time Optical Flow Hermite Transform*. In Nature-Inspired Computing for Control Systems, pages 231–263. Springer, 2016. (Cited in page 60.)
- [Muller 1987] Robert U Muller et John L Kubie. *The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells*. The Journal of Neuroscience, vol. 7, no. 7, pages 1951–1968, 1987. (Cited in page 34.)
- [Müller 2014] Stefan Müller, Cornelius Weber et Stefan Wermter. *RatSLAM on Humanoids-A Bio-Inspired SLAM Model Adapted to a Humanoid Robot*. In Artificial Neural Networks and Machine Learning–ICANN 2014, pages 789–796. Springer, 2014. (Cited in page 59.)
- [Murphy 2000] R Murphy. *Introduction to AI robots*, 2000. (Cité en pages xviii, 71, 72, 73 et 74.)
- [Nagel 1987] Hans-Hellmut Nagel. *On the estimation of optical flow: Relations between different approaches and some new results*. Artificial intelligence, vol. 33, no. 3, pages 299–324, 1987. (Cited in page 60.)
- [Nakhaeina 2011] Danial Nakhaeina, Sai Hong Tang, SB Mohd Noor et O Motlagh. *A review of control architectures for autonomous navigation of mobile robots*. International Journal of Physical Sciences, vol. 6, no. 2, pages 169–174, 2011. (Cited in page 67.)
- [Nattharith 2009] Panus Nattharith et Robert Bicker. *Mobile Robot Navigation using a Behavioural Strategy*. Control and Application (CA 2009). H. Hu. Cambridge, UK, 2009. (Cited in page 83.)
- [Neira 2001] José Neira et Juan D Tardós. *Data association in stochastic mapping using the joint compatibility test*. Robotics and Automation, IEEE Transactions on, vol. 17, no. 6, pages 890–897, 2001. (Cited in page 49.)
- [Newman 2002] Paul Newman, John Leonard, JD Tardó et José Neira. *Explore and return: Experimental validation of real-time concurrent mapping and localization*. In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, volume 2, pages 1802–1809. IEEE, 2002. (Cited in page 53.)
- [Nguyen 2013] Vu Anh Nguyen, Janusz A Starzyk et Wooi-Boon Goh. *A spatio-temporal Long-term Memory approach for visual place recognition in mobile robotic navigation*. Robotics and Autonomous Systems, vol. 61, no. 12, pages 1744–1758, 2013. (Cited in page 33.)

- [Nicolescu 2001] Monica N Nicolescu et Maja J Matarić. *Experience-based representation construction: learning from human and robot teachers*. In Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on, volume 2, pages 740–745. IEEE, 2001. (Cited in page 81.)
- [Nielsen 2005] Curtis W Nielsen, Michael Goodrich, Randall J Rupper *et al.* *Towards facilitating the use of a pan-tilt camera on a mobile robot*. In Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on, pages 568–573. IEEE, 2005. (Cited in page 47.)
- [Nilsson 1984] Nils J Nilsson. *Shakey the robot*. Rapport technique, DTIC Document, 1984. (Cited in page 72.)
- [Nister 2006] David Nister et Henrik Stewenius. *Scalable recognition with a vocabulary tree*. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), volume 2, pages 2161–2168. IEEE, 2006. (Cited in page 28.)
- [Nüchter 2007] Andreas Nüchter, Kai Lingemann, Joachim Hertzberg et Hartmut Surmann. *6D SLAM—3D mapping outdoor environments*. Journal of Field Robotics, vol. 24, no. 8-9, pages 699–722, 2007. (Cited in page 44.)
- [Nützi 2011] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza et Roland Siegwart. *Fusion of IMU and vision for absolute scale estimation in monocular SLAM*. Journal of intelligent & robotic systems, vol. 61, no. 1-4, pages 287–299, 2011. (Cited in page 45.)
- [Oh 2015] Jean H Oh, Arne Suppé, Felix Duvallet, Abdeslam Boularias, Luis E Navarro-Serment, Martial Hebert, Anthony Stentz, Jerry Vinokurov, Oscar J Romero, Christian Lebiere *et al.* *Toward Mobile Robots Reasoning Like Humans*. In AAAI, pages 1371–1379, 2015. (Cited in page 63.)
- [Ojala 2002] Timo Ojala, Matti Pietikäinen et Topi Mäenpää. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 7, pages 971–987, 2002. (Cited in page 63.)
- [O’Keefe 1971] John O’Keefe et Jonathan Dostrovsky. *The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat*. Brain research, vol. 34, no. 1, pages 171–175, 1971. (Cited in page 33.)
- [O’Keefe 1976] John O’Keefe. *Place units in the hippocampus of the freely moving rat*. Experimental neurology, vol. 51, no. 1, pages 78–109, 1976. (Cited in page 34.)
- [O’keefe 1978a] J O’keefe et DH Conway. *Hippocampal place units in the freely moving rat: why they fire where they fire*. Experimental Brain Research, vol. 31, no. 4, pages 573–590, 1978. (Cited in page 34.)
- [O’keefe 1978b] John O’keefe et Lynn Nadel. *The hippocampus as a cognitive map*. Oxford University Press, USA, 1978. (Cited in page 34.)

- [Oleynikova 2015] Helen Oleynikova, Michael Burri, Simon Lynen et Roland Siegwart. *Real-time visual-inertial localization for aerial and ground robots*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 3079–3085. IEEE, 2015. (Cited in page 56.)
- [Panzieri 2002] Stefano Panzieri, Federica Pascucci et Giovanni Ulivi. *An outdoor navigation system using GPS and inertial platform*. Mechatronics, IEEE/ASME Transactions on, vol. 7, no. 2, pages 134–142, 2002. (Cited in page 45.)
- [Parra 2010] Ignacio Parra, MA Sotelo, David Fernández Llorca et Manuel Ocaña. *Robust visual odometry for vehicle localization in urban environments*. Robotica, vol. 28, no. 03, pages 441–452, 2010. (Cited in page 52.)
- [Payton 1992] David W Payton, David Keirsey, Dan M Kimble, Jimmy Krozel et J Kenneth Rosenblatt. *Do whatever works: A robust approach to fault-tolerant autonomous control*. Applied Intelligence, vol. 2, no. 3, pages 225–250, 1992. (Cited in page 80.)
- [Perera 2011] Samunda Perera et Ajith Pasqual. *Towards realtime handheld MonoSLAM in dynamic environments*. In Advances in Visual Computing, pages 313–324. Springer, 2011. (Cited in page 56.)
- [Peretroukhin 2015] Valentin Peretroukhin, Lee Clement, Matthew Giamou et Jonathan Kelly. *PROBE: Predictive robust estimation for visual-inertial navigation*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 3668–3675. IEEE, 2015. (Cited in page 56.)
- [Peter Bonasso 1997] R Peter Bonasso, R James Firby, Erann Gat, David Kortenkamp, David P Miller et Mark G Slack. *Experiences with an architecture for intelligent, reactive agents*. Journal of Experimental & Theoretical Artificial Intelligence, vol. 9, no. 2-3, pages 237–256, 1997. (Cited in page 77.)
- [Pirjanian 1999] Paolo Pirjanian. *Behavior coordination mechanisms-state-of-the-art*. Rapport technique, Citeseer, 1999. (Cited in page 80.)
- [Qin 2013] Baoxing Qin, Zhuang Jie Chong, Tirthankar Bandyopadhyay et Marcelo H Ang. *Metric mapping and topo-metric graph learning of urban road network*. In 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), pages 119–123. IEEE, 2013. (Cited in page 48.)
- [Quirk 1990] Gregory J Quirk, Robert U Muller et John L Kubie. *The firing of hippocampal place cells in the dark depends on the rat’s recent experience*. The Journal of Neuroscience, vol. 10, no. 6, pages 2008–2017, 1990. (Cited in page 35.)
- [Qureshi 2004] Faisal Qureshi, Demetri Terzopoulos et Ross Gillett. *The cognitive controller: a hybrid, deliberative/reactive control architecture for autonomous robots*. In Innovations in Applied Artificial Intelligence, pages 1102–1111. Springer, 2004. (Cited in page 77.)

- [Ramaithitima 2016] Rattanachai Ramaithitima, Michael Whitzer, Subhrajit Bhattacharya et Vijay Kumar. *Automated Creation of Topological Maps in Unknown Environments Using a Swarm of Resource-Constrained Robots*. IEEE Robotics and Automation Letters, vol. 1, no. 2, pages 746–753, 2016. (Cited in page 48.)
- [Ramos 2012] Fabio Ramos, Ben Upcroft, Suresh Kumar et Hugh Durrant-Whyte. *A Bayesian approach for place recognition*. Robotics and Autonomous Systems, vol. 60, no. 4, pages 487–497, 2012. (Cited in page 33.)
- [Rasmussen 2014] Christopher Rasmussen, Yan Lu et Mehmet Kocamaz. *A trail-following robot which uses appearance and structural cues*. In Field and Service Robotics, pages 265–279. Springer, 2014. (Cited in page 21.)
- [Redish 1997] A David Redish et David S Touretzky. *Cognitive maps beyond the hippocampus*. Hippocampus, vol. 7, no. 1, pages 15–35, 1997. (Cited in page 33.)
- [Rencken 1993] Wolfgang D Rencken. *Concurrent localisation and map building for mobile robots using ultrasonic sensors*. In Intelligent Robots and Systems’ 93, IROS’93. Proceedings of the 1993 IEEE/RSJ International Conference on, volume 3, pages 2192–2197. IEEE, 1993. (Cited in page 53.)
- [Ribas 2008] David Ribas, Pere Ridao, Juan Domingo Tardós et José Neira. *Underwater SLAM in man-made structured environments*. Journal of Field Robotics, vol. 25, no. 11-12, pages 898–921, 2008. (Cited in page 44.)
- [Ribeiro 2015] Fernando Ribeiro, Susana Brandao, Joao P Costeira et Manuela Veloso. *Global localization by soft object recognition from 3D Partial Views*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 3709–3714. IEEE, 2015. (Cited in page 53.)
- [Rodríguez-Piñeiro 2011] José Rodríguez-Piñeiro, Pedro Comesaña-Alfaro, Fernando Pérez-González et Alberto Malvido-García. *A new method for perspective correction of document images*. In IS&T/SPIE Electronic Imaging, pages 787410–787410. International Society for Optics and Photonics, 2011. (Cited in page 140.)
- [Rolls 1999] Edmund T Rolls et al. *Spatial view cells and the representation of place in the primate hippocampus*. Hippocampus, vol. 9, no. 4, pages 467–480, 1999. (Cited in page 34.)
- [Rosen 2016] David M Rosen, Julian Mason et John J Leonard. *Towards Lifelong Feature-Based Mapping in Semi-Static Environments*. In International Conference on Robotics and Automation (ICRA). IEEE, 2016. (Cited in page 48.)
- [Rosenblatt 1997] Julio K Rosenblatt. *DAMN: A distributed architecture for mobile navigation*. Journal of Experimental & Theoretical Artificial Intelligence, vol. 9, no. 2-3, pages 339–360, 1997. (Cited in page 80.)

- [Ross 2008] David A Ross, Jongwoo Lim, Ruei-Sung Lin et Ming-Hsuan Yang. *Incremental learning for robust visual tracking*. International Journal of Computer Vision, vol. 77, no. 1-3, pages 125–141, 2008. (Cited in page 30.)
- [Rosten 2006] Edward Rosten et Tom Drummond. *Machine learning for high-speed corner detection*. In European conference on computer vision, pages 430–443. Springer, 2006. (Cited in page 27.)
- [Royer 2007] Eric Royer, Maxime Lhuillier, Michel Dhome et Jean-Marc Lavest. *Monocular vision for mobile robot localization and autonomous navigation*. International Journal of Computer Vision, vol. 74, no. 3, pages 237–260, 2007. (Cited in page 45.)
- [Ruble 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige et Gary Bradski. *ORB: An efficient alternative to SIFT or SURF*. In 2011 International conference on computer vision, pages 2564–2571. IEEE, 2011. (Cited in page 28.)
- [Rusiñol 2010] Marçal Rusiñol et Josep Lladós. Symbol spotting in digital libraries: Focused retrieval over graphic-rich document collections. Springer Science & Business Media, 2010. (Cited in page 142.)
- [Rusinol 2013] Marçal Rusinol, Dimosthenis Karatzas et Josep Lladós. *Spotting graphical symbols in camera-acquired documents in real time*. In International Workshop on Graphics Recognition, pages 3–10. Springer, 2013. (Cited in page 142.)
- [Saffiotti 1997] Alessandro Saffiotti. *The uses of fuzzy logic in autonomous robot navigation*. Soft Computing, vol. 1, no. 4, pages 180–197, 1997. (Cited in page 80.)
- [Sala 2006] Pablo Sala, Robert Sim, Ali Shokoufandeh et Sven Dickinson. *Landmark selection for vision-based navigation*. Robotics, IEEE Transactions on, vol. 22, no. 2, pages 334–349, 2006. (Cited in page 61.)
- [Sarkar 2005] Anjan Sarkar, Anjan Banerjee, Nilanjan Banerjee, Siddhartha Brahma, B Kartikeyan, Manab Chakraborty et Kantilal L Majumder. *Landcover classification in MRF context using Dempster-Shafer fusion for multisensor imagery*. Image Processing, IEEE Transactions on, vol. 14, no. 5, pages 634–645, 2005. (Cited in page 45.)
- [Schmid 2000] Cordelia Schmid, Roger Mohr et Christian Bauckhage. *Evaluation of interest point detectors*. International Journal of computer vision, vol. 37, no. 2, pages 151–172, 2000. (Cited in page 27.)
- [Schmiedel 2015] Thomas Schmiedel, Erik Einhorn et Horst-Michael Gross. *IRON: A fast interest point descriptor for robust NDT-map matching and its application to robot localization*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 3144–3151. IEEE, 2015. (Cited in page 54.)
- [Schulz 2015] Ruth Schulz, Ben Talbot, Obadiah Lam, Feras Dayoub, Peter Corke, Ben Ugcroft et Gordon Wyeth. *Robot navigation using human cues: a robot navigation system for symbolic goal-directed exploration*. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 1100–1105. IEEE, 2015. (Cited in page 49.)

- [Schuermans 2007] Li Cheng SVN Schuermans et SW Caelli. *Implicit online learning with kernels*. Advances in neural information processing systems, vol. 19, page 249, 2007. (Cited in page 31.)
- [Schwartz 1983] Jacob T Schwartz et Micha Sharir. *On the “piano movers” problem. II. General techniques for computing topological properties of real algebraic manifolds*. Advances in applied Mathematics, vol. 4, no. 3, pages 298–351, 1983. (Cited in page 72.)
- [Schweighofer 2006] Gerald Schweighofer et Axel Pinz. *Robust pose estimation from a planar target*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 28, no. 12, pages 2024–2030, 2006. (Cited in page 62.)
- [Se 2005] Stephen Se, David G Lowe et James J Little. *Vision-based global localization and mapping for mobile robots*. Robotics, IEEE Transactions on, vol. 21, no. 3, pages 364–375, 2005. (Cited in page 55.)
- [Shaikh 2013] Sayyan N Shaikh et Neelakantha V Londhe. *OCR Based Mapless Navigation Method of Robot*. International Journal of Inventive Engineering and Sciences, vol. 1, no. 8, pages 6–12, 2013. (Cited in page 63.)
- [Sharif Razavian 2014] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan et Stefan Carlsson. *CNN features off-the-shelf: an astounding baseline for recognition*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 806–813, 2014. (Cited in page 33.)
- [Shotton 2009] Jamie Shotton, John Winn, Carsten Rother et Antonio Criminisi. *Textronboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context*. International Journal of Computer Vision, vol. 81, no. 1, pages 2–23, 2009. (Cited in page 63.)
- [Siagian 2014] Christian Siagian, Chin Kai Chang et Laurent Itti. *Autonomous mobile robot localization and navigation using a hierarchical map representation primarily guided by vision*. Journal of Field Robotics, vol. 31, no. 3, pages 408–440, 2014. (Cited in page 48.)
- [Singh 2010] Krishna Kant Singh et Akansha Singh. *A study of image segmentation algorithms for different types of images*. International Journal of Computer Science, vol. 7, no. 5, pages 414–417, 2010. (Cited in page 136.)
- [Skinner 1974] B. F. Skinner. *Behaviorism at fifty*. New York, J. Norton Publishers, 1974. (Cited in page 79.)
- [Smith 1986] Randall C Smith et Peter Cheeseman. *On the representation and estimation of spatial uncertainty*. The international journal of Robotics Research, vol. 5, no. 4, pages 56–68, 1986. (Cited in page 53.)

- [Smith 1990] Randall Smith, Matthew Self et Peter Cheeseman. *Estimating uncertain spatial relationships in robotics*. In Autonomous robot vehicles, pages 167–193. Springer, 1990. (Cited in page 55.)
- [Sobel 1968] Irwin Sobel et Gary Feldman. *A 3x3 isotropic gradient operator for image processing*. a talk at the Stanford Artificial Project in, pages 271–272, 1968. (Cited in page 27.)
- [Sola 2007] Joan Sola. *Multi-camera VSLAM: from former information losses to self-calibration*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Diego, CA, USA. Citeseer, 2007. (Cited in page 55.)
- [Sotelo 2007] Miguel-angel Sotelo, Roberto García, Ignacio Parra, David Fernández, Miguel Gavilán, Sergio Álvarez et José-eugenio Naranjo. *Visual odometry for road vehicles—feasibility analysis*. Journal of Zhejiang University SCIENCE A, vol. 8, no. 12, pages 2017–2020, 2007. (Cited in page 52.)
- [Strasdat 2011] Hauke Strasdat, Andrew J Davison, JMM Montiel et Kurt Konolig. *Double window optimisation for constant time visual SLAM*. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 2352–2359. IEEE, 2011. (Cited in page 55.)
- [Strasdat 2012] Hauke Strasdat, José MM Montiel et Andrew J Davison. *Visual SLAM: why filter?* Image and Vision Computing, vol. 30, no. 2, pages 65–77, 2012. (Cited in page 58.)
- [Sun 2001a] Ron Sun. *Duality of the mind: A bottom-up approach toward cognition*. Psychology Press, 2001. (Cited in page 12.)
- [Sun 2001b] Ron Sun, Edward Merrill et Todd Peterson. *From implicit skills to explicit knowledge: A bottom-up model of skill learning*. Cognitive science, vol. 25, no. 2, pages 203–244, 2001. (Cited in page 12.)
- [Suzuki 1985] Satoshi Suzukiet al. *Topological structural analysis of digitized binary images by border following*. Computer Vision, Graphics, and Image Processing, vol. 30, no. 1, pages 32–46, 1985. (Cited in page 137.)
- [Syed 1999] Nadeem Ahmed Syed, Huan Liu et Kah Kay Sung. *Handling concept drifts in incremental learning with support vector machines*. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 317–321. ACM, 1999. (Cited in page 30.)
- [Takahashi 1989] Osamu Takahashi et Robert J Schilling. *Motion planning in a plane using generalized Voronoi diagrams*. Robotics and Automation, IEEE Transactions on, vol. 5, no. 2, pages 143–150, 1989. (Cited in page 72.)
- [Tardós 2002] Juan D Tardós, José Neira, Paul M Newman et John J Leonard. *Robust mapping and localization in indoor environments using sonar data*. The International Journal of Robotics Research, vol. 21, no. 4, pages 311–330, 2002. (Cited in page 44.)

- [Tejera 2013] Gonzalo Tejera, Alejandra Barrera, Martin Llofriu et Alfredo Weitzenfeld. *Solving uncertainty during robot navigation by integrating grid cell and place cell firing based on rat spatial cognition studies*. In Advanced Robotics (ICAR), 2013 16th International Conference on, pages 1–6. IEEE, 2013. (Cited in page 35.)
- [Thorndike 1913] Edward L Thorndike. *Educational psychology, Vol 2: The psychology of learning*. 1913. (Cited in page 79.)
- [Thrun 2005] Sebastian Thrun, Wolfram Burgard et Dieter Fox. Probabilistic robotics. MIT press, 2005. (Cited in page 52.)
- [Thrun 2006] Sebastian Thrun et Michael Montemerlo. *The graph SLAM algorithm with applications to large-scale mapping of urban structures*. The International Journal of Robotics Research, vol. 25, no. 5-6, pages 403–429, 2006. (Cited in page 44.)
- [Thrun 2008] Sebastian Thrun et John J Leonard. *Simultaneous localization and mapping*. In Springer handbook of robotics, pages 871–889. Springer, 2008. (Cité en pages 55 et 56.)
- [Tipaldi 2013] Gian Diego Tipaldi, Daniel Meyer-Delius et Wolfram Burgard. *Lifelong localization in changing environments*. The International Journal of Robotics Research, vol. 32, no. 14, pages 1662–1678, 2013. (Cited in page 57.)
- [Tolman 1948] Edward C Tolman. *Cognitive maps in rats and men*. Psychological review, vol. 55, no. 4, page 189, 1948. (Cited in page 33.)
- [Torii 2013] Akihiko Torii, Josef Sivic, Tomas Pajdla et Masatoshi Okutomi. *Visual place recognition with repetitive structures*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 883–890, 2013. (Cité en pages 28 et 32.)
- [Ulanovsky 2007] Nachum Ulanovsky et Cynthia F Moss. *Hippocampal cellular and network activity in freely moving echolocating bats*. Nature neuroscience, vol. 10, no. 2, pages 224–233, 2007. (Cité en pages 34 et 35.)
- [Ullah 2008] Muhammad Muneeb Ullah, Andrzej Pronobis, Barbara Caputo, Jie Luo, Patric Jensfelt et Henrik I Christensen. *Towards robust place recognition for robot localization*. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 530–537. IEEE, 2008. (Cited in page 33.)
- [Valiente 2015] David Valiente, Maani Ghaffari Jadidi, Jaime Valls Miró, Arturo Gil et Oscar Reinoso. *Information-based view initialization in visual SLAM with a single omnidirectional camera*. Robotics and Autonomous Systems, 2015. (Cited in page 46.)
- [Vardy 2003] Andrew Vardy et Franz Oppacher. *Low-level visual homing*. In Advances in Artificial Life, pages 875–884. Springer, 2003. (Cited in page 62.)
- [Wang 2012] Yin-Tien Wang, Chung-Hsun Sun et Ming-Jang Chiou. *Detection of moving objects in image plane for robot navigation using monocular vision*. EURASIP Journal

- on Advances in Signal Processing, vol. 2012, no. 1, pages 1–22, 2012. (Cited in page 46.)
- [Warren 2014] Michael Warren, David McKinnon, Hu He, Arren Glover, Michael Shiel et Ben Upcroft. *Large scale monocular vision-only mapping from a fixed-wing sUAS*. In Field and Service Robotics, pages 495–509. Springer, 2014. (Cited in page 56.)
- [Weber 2012] Jonathan Weber et Salvatore Tabbone. *Symbol spotting for technical documents: An efficient template-matching approach*. In ICPR, pages 669–672, 2012. (Cited in page 142.)
- [Wehner 1979] R Wehner et F Räber. *Visual spatial memory in desert ants, Cataglyphis bicolor (Hymenoptera: Formicidae)*. Experientia, vol. 35, no. 12, pages 1569–1571, 1979. (Cited in page 25.)
- [Wiener 1961] Norbert Wiener. Cybernetics or control and communication in the animal and the machine, volume 25. MIT press, 1961. (Cited in page 71.)
- [Williams 2008] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid et Juan Tardós. *An image-to-map loop closing method for monocular SLAM*. In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 2053–2059. IEEE, 2008. (Cited in page 56.)
- [Wu 2009] Jianxin Wu, Henrik I Christensen et James M Rehg. *Visual place categorization: Problem, dataset, and algorithm*. In 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4763–4770. IEEE, 2009. (Cited in page 33.)
- [Xiang 2015] Lingzhu Xiang, Zhile Ren, Mengrui Ni et Odest Chadwicke Jenkins. *Robust graph SLAM in dynamic environments with moving landmarks*. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 2543–2549. IEEE, 2015. (Cited in page 57.)
- [Yartsev 2011] Michael M Yartsev, Menno P Witter et Nachum Ulanovsky. *Grid cells without theta oscillations in the entorhinal cortex of bats*. Nature, vol. 479, no. 7371, pages 103–107, 2011. (Cited in page 35.)
- [Yartsev 2013] Michael M Yartsev et Nachum Ulanovsky. *Representation of three-dimensional space in the hippocampus of flying bats*. Science, vol. 340, no. 6130, pages 367–372, 2013. (Cited in page 35.)
- [Yavuz 2002] H Yavuz et A Bradshaw. *A new conceptual approach to the design of hybrid control architecture for autonomous mobile robots*. Journal of Intelligent and Robotic Systems, vol. 34, no. 1, pages 1–26, 2002. (Cited in page 77.)
- [Yun 2008] Jooseop Yun et Jun Miura. *A quantitative measure for the navigability of a mobile robot using rough maps*. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3458–3464. IEEE, 2008. (Cited in page 49.)

- [Zhang 2012] Chi Zhang, Jing Xu, Ning Xi, Yunyi Jia et Weixian Li. *Development of an omnidirectional 3D camera for robot navigation*. In Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on, pages 262–267. IEEE, 2012. (Cited in page 46.)
- [Zhang 2015] Qiwen Zhang, Ioannis Rekleitis et Gregory Dudek. *Uncertainty reduction via heuristic search planning on hybrid metric/topological map*. In Computer and Robot Vision (CRV), 2015 12th Conference on, pages 222–229. IEEE, 2015. (Cited in page 48.)
- [Zhou 2003] Chao Zhou, Yucheng Wei et Tieniu Tan. *Mobile robot self-localization based on global visual appearance features*. In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, volume 1, pages 1271–1276. IEEE, 2003. (Cited in page 62.)
- [Zingg 2010] Simon Zingg, Davide Scaramuzza, Stephan Weiss et Roland Siegwart. *MAV navigation through indoor corridors using optical flow*. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pages 3361–3368. IEEE, 2010. (Cited in page 60.)

Appendix

A.1 Rhizome Architecture Optimization

A.1.1 Lost Sign Searching Reflex Behavior (LSSRB)

In order to overcome the problem related to the loss of sight of the current sign due to lighting variation or other reason, and additional reflex behavior layer was integrated to the already in-built layers of the behavioral module.

The overall layer is composed of seven neural groups which some are connected to some groups of the three already built-in layers as illustrated in figure A.9. Therefore, it has not input units.

The layer is in charge of controlling a reflex movement of the robot whenever it loses the sight of the current sign. This case is prone to happen when the robot is approaching the sign. As explained in the TARB layer, the robot approaches it by performing some left, right or straight ahead movements allowing it to have the sign centered within its field of view. However, at some point, it is possible that the sign gets out of view after the performance of any of these movements.

Therefore, in order to look again for the same sign, a logical solution is here considered. In fact, by performing the exact opposite movement where the robot last saw the sign, there is a high probability of finding it again. To this end, it is necessary to keep in memory the activity of the performed movements at every perception-action cycle. This is done in the *Memory position group*.

Additionally, it is necessary to know that the sign has not been recognized at all in order to start searching for it. Therefore, the layer is connected to the *sign recognition detector* group of the DDRB layer whose deactivation (sign not recognized) triggers the activation of the search movements. However, since the lack of a sign in the environment could also mean that the robot is looking for the next expected sign (see DDRB layer), it is important for the robot to be able to distinguish each situation.

This is possible by using the *sign lost memory group* storing the information of the current perceived sign until the robot has achieved it through the connection of two groups of the SRMA layer. Hence, if the sign stored in the *short memory group* and transferred to the *WTA group* is the same as the expected sign at the time there is no sign recognized in the environment, then the sign is considered lost on the way.

Otherwise, if the expected sign is different from the stored, it would simply mean that the robot is looking for the next expected sign after having attained the current one.

Internal units

Sign Lost Memory group: It stores the current expected sign that it is being perceived by the robot. It has two inputs coming from the SRMA layer: on one side, the *sign sequence*

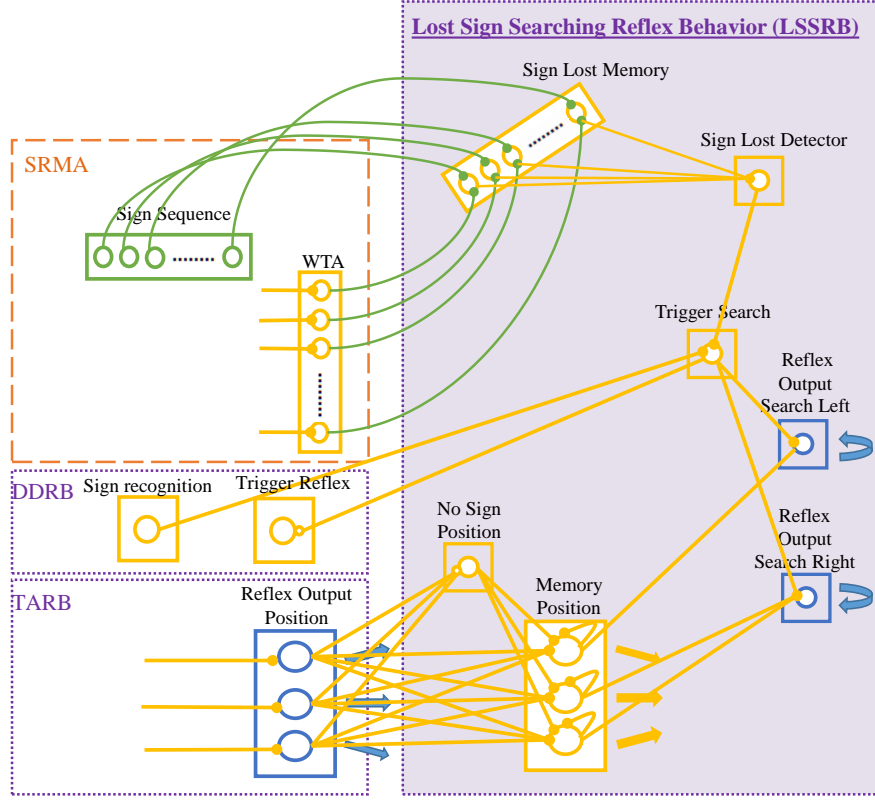


Figure A.1: Detailed view of the Lost Sign Searching Reflex Behavior (LSSRB) layer composed of seven neural groups.

group indicating the expected sign to be found and on the other side the first *WTA group* indicating the sign neuron that is currently being perceived. By considering a threshold allowing this group to be activated only when both inputs are activated, it is possible to know that the robot has lost or not the sign from its field of view. If it has indeed lost it, it is possible to still know which sign it was previously looking for as it has been kept in memory.

Sign Lost Detector group: It consists of a single neuron serving, as its name indicates it, as a detector of a lost sign regardless of the sign. The activation of a single neuron is sufficient to activate this group.

A zoom view of these two groups in the layer is illustrated in figure A.2.

Trigger search group: This neural group is essential in the functionality of the whole layer as it is in charge of triggering the activation of the *reflex output search (left or right) groups* when it is necessary. It receives as input the information coming from the *sign lost detector group* and the *sign recognition group*. This latter is linked by an inhibitory connection allowing the activation of the *trigger search group* when no sign has been recognized which itself inhibits the activation of the reflex movements of the DDRB layer through the *trigger reflex group* (see figure A.3).

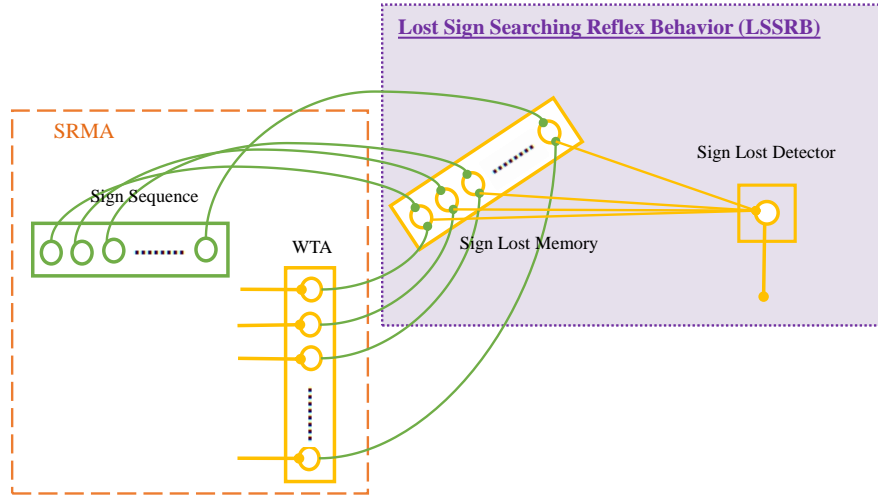


Figure A.2: Activity and connection of the Sign Lost Memory group connected to the Sign sequence group of the SRMA layer and the Sign Lost detector group connected to the first WTA group of the SRMA layer.

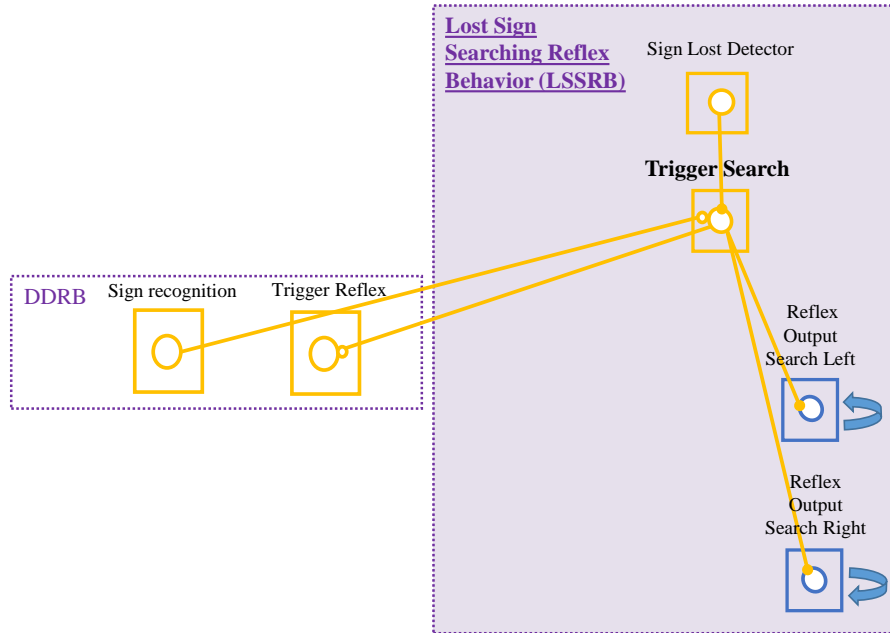


Figure A.3: Activity of the trigger search group. When activated, it inhibits the activity of the reflex movement performed by the DDRB.

Memory Position group: This group is composed of the same number of neurons as the *reflex output position* of the TARB layer to which it is connected. They correspond to

the three possible movements allowing the robot to walk while approaching the sign: right, straight ahead and left. The connection is made such that it keeps in memory the most recent movement performed by the robot while walking but in the neuron corresponding to the opposite side (i.e. left neuron in the *reflex output position* activates the right neuron in this *memory position group*). The straight-ahead movement neuron remains intact (see figure A.4).

NoSignPosition group: This group allows triggering the activation of the corresponding neuron in the *memory position group* only when no sign is recognized at all within the robot's field of view. In fact, since it is connected to all three neurons of the *reflex output position* via an inhibitory link, it is possible to detect whether a sign, wherever it is positioned within the robot's field of view, is recognized or not.

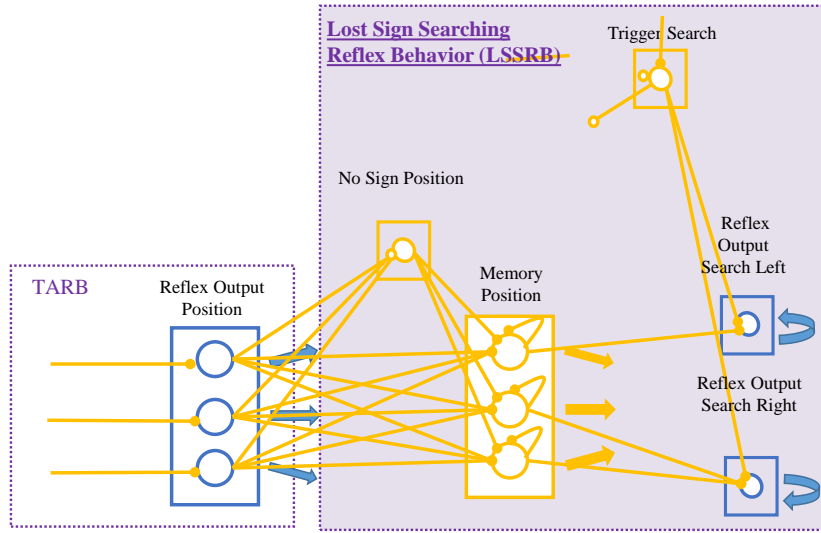


Figure A.4: Activity of the Memory Position group and its connection to the output units. The position is stored according to the last opposed movement. For instance if the robot last movement was to the right, then the movement to be performed to find again the sign would be to the left and Vice-versa

Output units

Reflex Output Search Left group: This single-neuron group allows the robot to perform a rotatory reflex movement to the left whenever the last performed robot movement was performed to the opposite side (right).

Reflex Output Search Right group: This single-neuron group allows the robot to perform a rotatory reflex movement to the left whenever the last performed robot movement was performed to the opposite side (to the left) as well as straight ahead.

Both Reflex Output search groups are linked to the Motor Output in order to execute the movements (left or right). The reflex movement to the left being already encoded in the

Motor Output, a simple excitatory connection linking the corresponding neuron to the reflex output search left is enough. Moreover, a new neuron is added in the Motor Output group in order to encode the right reflex movement triggered by the Reflex Output Search Right group. Figure A.5 illustrates the convergence of this layer in the Motor Output group holding an additional neuron movement.

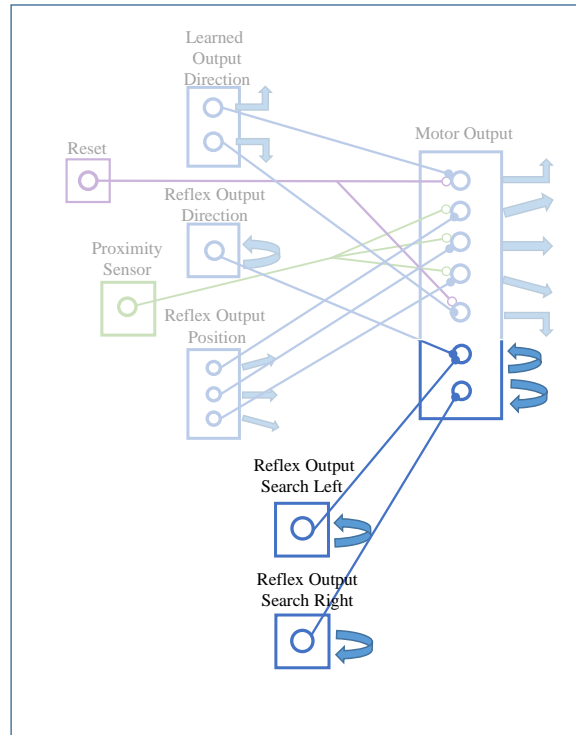


Figure A.5: Convergence of both reflexoutput Search groups (left and right) in the Motor Output (MO) group. A Reflex movement to the right neuron has been added to the final Motor Output(MO)

Finally, in order to be coherent with the rest of the architecture a new neural group **Reset Memory** group was added and the connection of the other reset groups was readjusted as illustrated in figure A.6.

The complete architecture as illustrated in the introduction is depicted in figure A.7 with the new **Lost Sign Searching Reflex layer**:

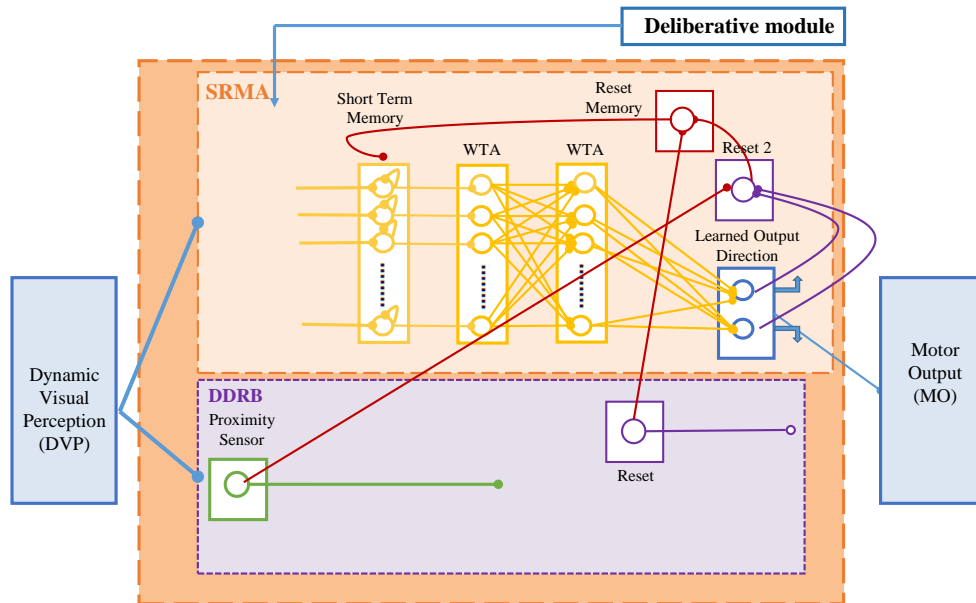


Figure A.6: Connectivity of the three reset neuron groups used in the overall architecture. Here, the third reset Memory has been added

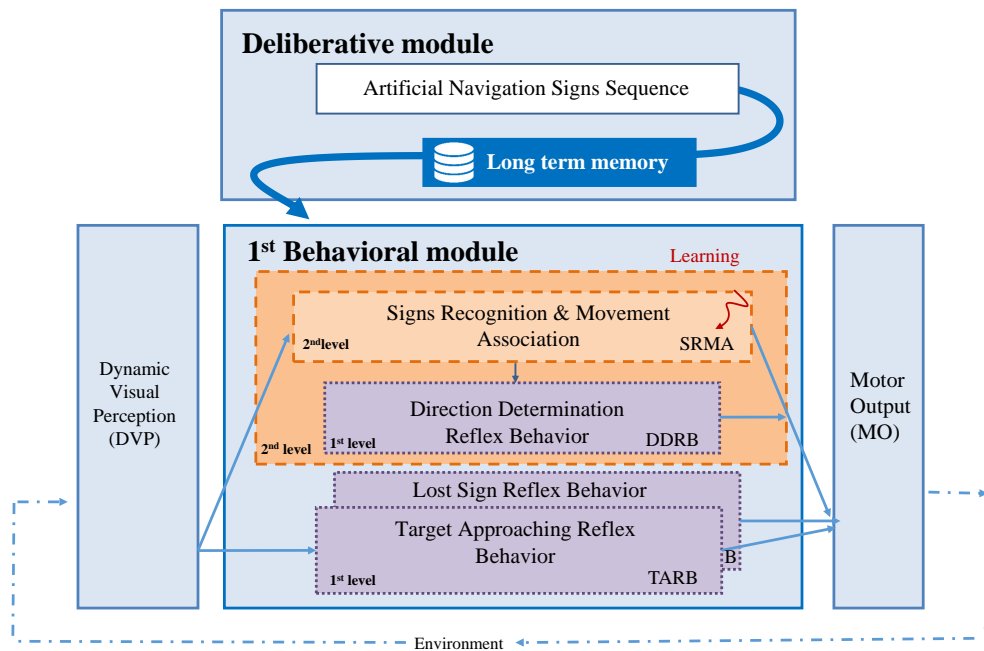


Figure A.7: General view of the complete Architecture. Rhizome 1 is composed of two modules, one deliberative and one behavioral. This latter is composed of three reflex behaviors layers and one recognition layer

A.2 TOOLBOX

A.2.1 SIFT : Scale-Invariant Feature Transform

The SIFT algorithm detects and computes a description of the interest points or so-called key points in the SIFT framework. A key point is defined by its (x, y) coordinates within the image and by its characteristic scaling factor (σ). A key point is a circular region of interest which area radius is proportional to the scale factor.

The complete process follows mainly four steps that can be found in more detailed in [Lowe 2004]:

A.2.1.1 Scale-space Extrema Detection

First, the problem of image scaling is solved by using a filter in a discrete space called scale space of three dimensions (x, y, σ) . It uses Difference of Gaussians (DOG) in order to find the local maxima across the scale and space. As result, it enhances the precision about the location of detected key points while eliminating a number considered irrelevant.

To that end, the Gaussian-smooth image L results from the convolution between the original image I and the Gaussian filter G of parameter σ as shown in the equation.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

After this operation, the image is smooth and all key points with a radius inferior to sigma are eliminated. Then the detection of key points of dimension approximately equal to σ is done by studying the image with difference of Gaussians with two different σ , let it be σ and $k\sigma$, defined as follows:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

As a result, only the objects that persist in the scale factors that vary between σ and $k\sigma$ are observable. Therefore, a key point candidate (x, y, σ) is defined as a point where an extremum of the DOG is achieved in relation to its immediate neighbors, that is to say the whole set composed of 26 points. For instance, one pixel in an image is compared with its eight neighbors pixels of the same scale and the eighteen pixels of the next and previous scale (nine and nine respectively). Then, if it is a local extrema, it is a potential keypoint, which is best represented in that scale.

A.2.1.2 Keypoint Localization

In order to have accurate results about the keypoints, another process is applied on the potential keypoints found in the previous stage. In fact, not all of them are quite stable and their localization might be not be accurate. Therefore, by using taylor series expansion of scale space, all low-contrast keypoints and edge keypoints are eliminated.

A.2.1.3 Orientation Assignment

This step consist of assigning to each remaining keypoint, one or several orientations depending only on the local content of the image in the vicinity of the key point at the given scale factor. This stage is essential to ensure the invariance of the rotation. Hence, the same descriptors should be able to be obtained from any image whatever its orientation is. To that end, the gradient magnitude $m(x, y)$ and direction $\theta(x, y)$ is calculated for each pixel in the neighborhood of a given keypoint (x_0, y_0, σ_0) of the Gaussien-smoothed image $L(x, y, \sigma)$ defined as follows:

$$\begin{cases} m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \\ \theta(x, y) = \arctan(L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y)) \\ \forall (x, y) \text{ in the vicinity of } (x_0, y_0) \end{cases}$$

Then, an orientation histogram with 36 bins covering 360 degrees is created. The bin with the highest value in the histogram is considered as the dominant orientation and if any other of them which value exceeds the 80 % of the dominant one, it is then use to calculate other interest point with the same position and scale but with a different orientation.

At the end of this stage, the keypoint is defined by four parameters: $(x_0, y_0, \sigma_0, \theta)$.

A.2.1.4 Keypoint Descriptor

Once the keypoint has been detected and its scale and rotation invariance has been ensured, a unique signature associated to each keypoint is computed. To that end, a whole region around the keypoint of 4×4 zones of 4×4 pixels each is considered. Then, for each zone, an orientation histogram with 8 bins corresponding to 8 directions is computed (A.8). The values added to each histogram bin are given by the gradient magnitude and orientation. As a result, the 16 histograms each with 8 bins are stored in a vector to finally provide the keypoint SIFT descriptor of 128 dimensions. The high dimensionality describing a keypoint makes of it a highly distinctive point invariant to the illumination, 3D viewpoint changes and other minor variations besides the already mentioned scale and rotation variations. Figure A.8 depicts the orientation histogram over 8 directions for each zone.

A.2.2 Bag of visual words

The bag of visual words model consist of two steps as shown in figure 9. First, the construction of a vocabulary allowing to identify the visual words by clustering the whole set of features is extracted (1.a) according to their similarity (1.b). Second, the assignment of features of the new images to the cluster with the closest centroid (2.a), followed by the histogram of the number of occurrences of the features in the given image (2.b). The main advantages of this method is its simplicity, its computational efficiency and its invariance to affine transformation, occlusion, lighting and intra class variation. Therefore, even though the complete process converges upon the categorization of images, our method uses the same process with a slight modification at the end, but instead, to converge upon the creation of

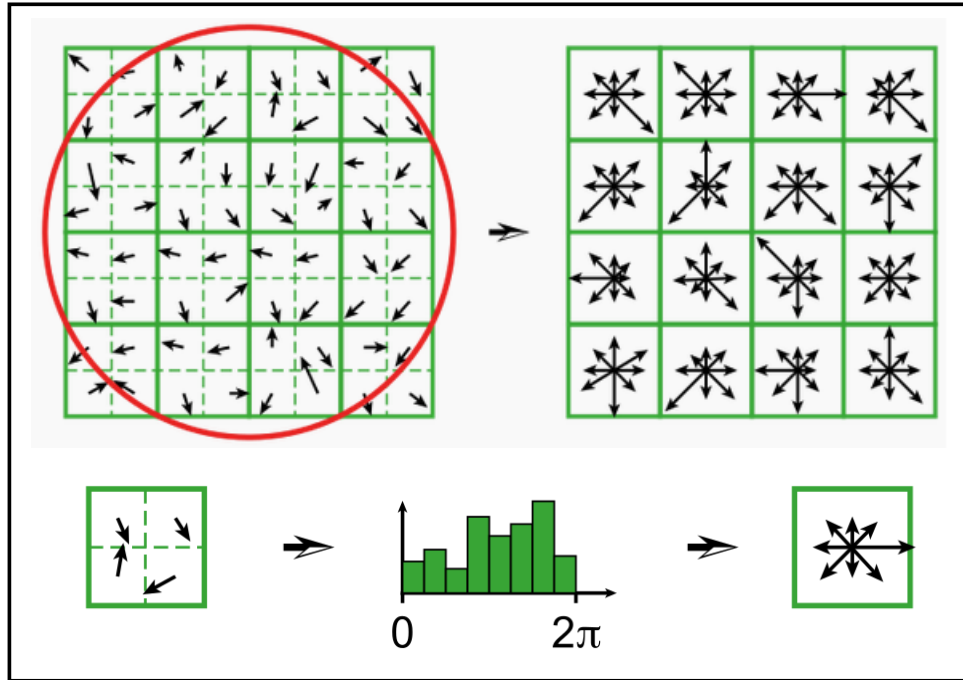


Figure A.8: SIFT descriptor construction with its corresponding histogram.

natural landmarks characterized by the presence or absence of some visual words, the detailed description of which is given in the implementation part (section 4.2.2.4).

A.2.2.1 Vocabulary construction

The construction of the vocabulary requires a quantization of the representation space of the local descriptors. More precisely, it requires the construction of a function of the representation space (Space in 128 dimensions in our case) towards a discrete space of labels. The visual words of a vocabulary are created to model the local descriptors from the query images. Therefore, the vocabulary must be adapted to the images that are to be processed. To this end, a set of template images is necessary. The most common method to build a vocabulary consists of extracting the descriptors from the template images and finding similarity among them. Then, by means of clustering algorithms, all similar descriptors are grouped into a k number of clusters. The total number of clusters is the size of the vocabulary and each of them has a center descriptor, which represents all the descriptors belonging to that cluster. Even though it is the most representative descriptor, it does not necessarily mean that the descriptors exist within the cluster, it might just be computed as the closest representation of all the other descriptors. Most of the methods proposed to quantify the local representations, use the k -means algorithm applied to local descriptors. Indeed, this combination of the use of feature detectors and the k -means algorithm for clustering has become very popular in the representation of bag-of-words.

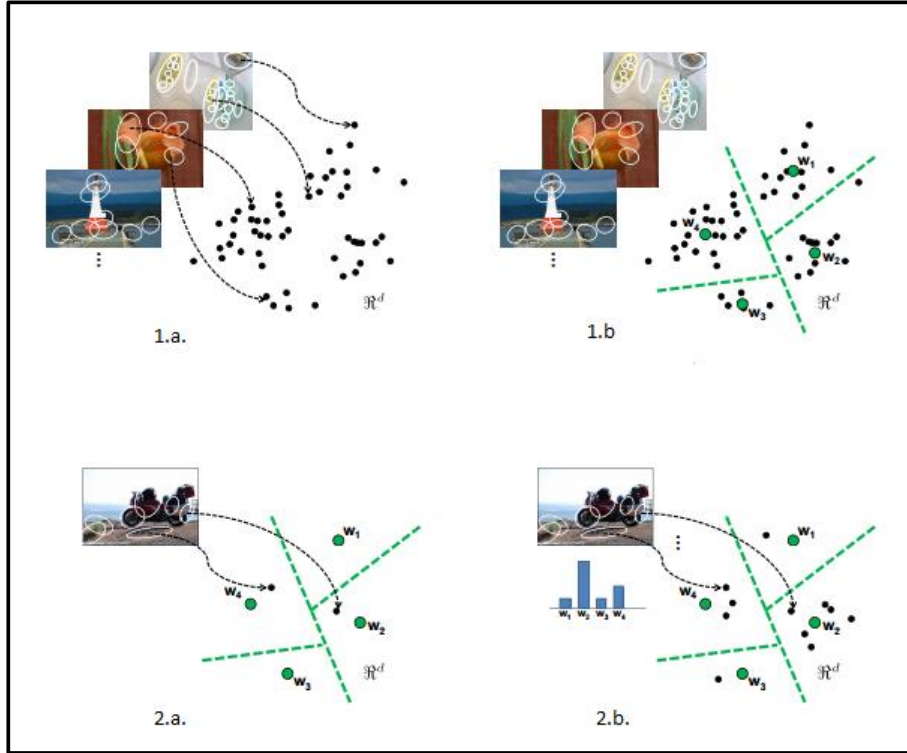


Figure A.9: Bag-of-visual words. First, the construction of a vocabulary allowing to identify the visual words by clustering the whole set of features is extracted (1.a) according to their similarity (1.b). Second, the assignment of features of the new images to the cluster with the closest centroid (2.a), followed by the histogram of the number of occurrences of the features in the given image (2.b).

A.2.2.2 K-means algorithm for clustering

K-means is an unsupervised learning algorithm that classifies a given set of data into a certain number of k clusters fixed in advance. It consists of mainly three steps. At first, it randomly chooses k samples (here descriptors) from the given data and assigns each of them to each created cluster. The clusters are the seed classes and the chosen samples are the centroids. Once the centroids initialization is over, the two following steps are performed in a loop: First, the distance between each sample and the centroids is measured and whenever it is minimal, the sample is assigned to the class comprising the nearest centroid. Then, new centroids are computed as barycenters of the clusters computed in the previous step by taking the mean value of all the samples belonging to the corresponding class. As a result, the k centroids change their positions after repetition of the last two steps. Thereafter, the loop goes on until the position of the resulting k centroids do not change significantly with respect to the last computed centroids. Thereafter, the same label is given to represent each class and its

corresponding elements. The distance is computed as the sum of the squared error:

$$\sum_{i=0}^n \min (\|x_j - c_i\|^2)$$

x_j	data samples
c_i	the centroid
$\ x_j - c_i\ $	the distance between a data sample

In our case, since the samples are given by the descriptors in 128 dimensions, the distance is computed by comparing all 128 values. The k -means algorithm will always converge after several iterations. However, it might only be a local minimum. The result highly relies on the initialization of the centroids. Therefore, it usually chooses the initial centroids distant from each other and the computation is performed many times by using different initializations of the centroids to finally choose the best one.

A.2.2.3 Image representation by Histogram

Once the vocabulary has been built, all new images can be described based on the same visual keypoints. To this end, the query images go under the same process of feature extraction described above and the Euclidean distance is computed between each descriptor and the final centroids of the vocabulary. Thus, the new descriptors are assigned to the class corresponding to the nearest centroid. Thereafter, the image can be represented by the histogram, which is computed by counting the number of descriptors assigned to each class. The number of class correspond to the total number of clusters in the vocabulary and bins in the histogram.

A.2.3 Adaptive Resonance Theory (ART)

The ART system represents a family of neural networks which uses supervised and unsupervised learning methods that cope with the plasticity-stability dilemma when addressing the problem of pattern recognition. The theory develops some aspects of the learning process computed by the human brain. Indeed, the incoming information is stored in clusters or categories that are constantly modified according to the new input elements. This occurs as a result of the comparison between the learned prototype and the sensory information. It is said that the system “resonates” when a category prototype resembles sufficiently to the current input vector and it is then when the learning takes place. Thus, only when the system is in a resonant state it can learn.

The basic ART is based on an unsupervised model and has a self-regulating control structure that allows a stable autonomous recognition and learning. It is mainly composed of four components: a comparison vector field, a recognition field, a vigilance parameter and a reset module as seen in figure A.10. Both, the *comparison* and *recognition* fields are composed of a set of neurons encoding respectively the input vectors and the category to which the input vectors are classified. The *vigilance parameter* works as a threshold of similarity between the input vectors and the categories and *the reset module* compares the threshold value to the strength of the recognition match after the input vectors are classified.

The value of the “vigilance parameter” is quite essential on the recognition task. The memory can be refined or generalized depending on the chosen value. Hence, a higher value produce the creation of many categories whereas a low value results in fewer categories.

Consequently, whenever the comparison field receives an input vector, it transfers it to the neuron in the recognition field whose set of weight matches the best to the input. Then, a lateral inhibition process takes places in the recognition field and only one neuron is activated. Subsequently, its resulting match value is compared to the vigilance term and only if it is within the normal range, training can take place by adjusting the weights of the firing recognition neuron towards the features of the input vector. Otherwise, if the comparison is below the threshold value, the firing recognition neuron is inhibited and a new cluster neuron is created in the recognition field and its weights are adjusted to match the input vector.

Since, new categories can be formed when the input vector does not match the information previously learned without modifying the stored input vectors unless they are sufficiently similar, the ART network has both plasticity and stability and can be used as model in the context of our work.

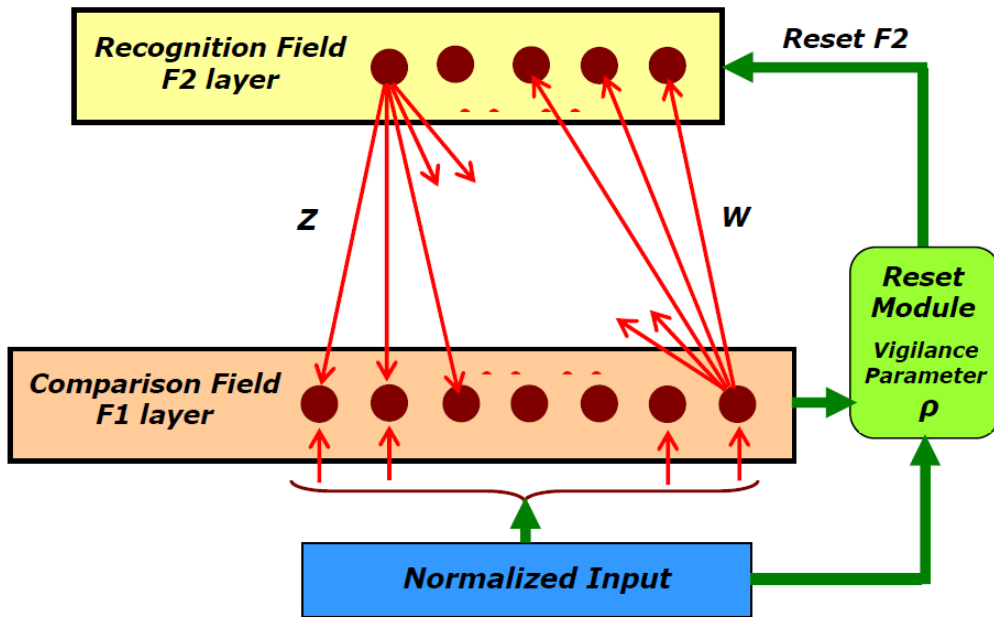


Figure A.10: ART Adaptive Resonance Theory

A.2.4 Object recognition

Depending on the approach or the desired task, different methods can be used allowing object recognition. Over the past decades, several methods have been proposed in the literature giving a big variety of choices with respect to the application. For instance, [Grauman 2011]

introduce in their book “Visual Object recognition” some of the most common algorithms for object recognition based on the advances in the computer vision literature of the last decade or so. They structure their lecture according to two types of recognition. The specific type, where the aim is to recognize an instance of a particular object and the generic category, which aims at recognizing different instances of a category that belongs to the same conceptual class.

Even though this distinction is important, mainly when working with image categorization, we have observed that the algorithms used for the specific case are usually also used for the generic one. For further details, the reader can refer to the literature mentioned in figure A.11. The list being non-exhaustive the reader can also refer to object recognition outside references for more details.

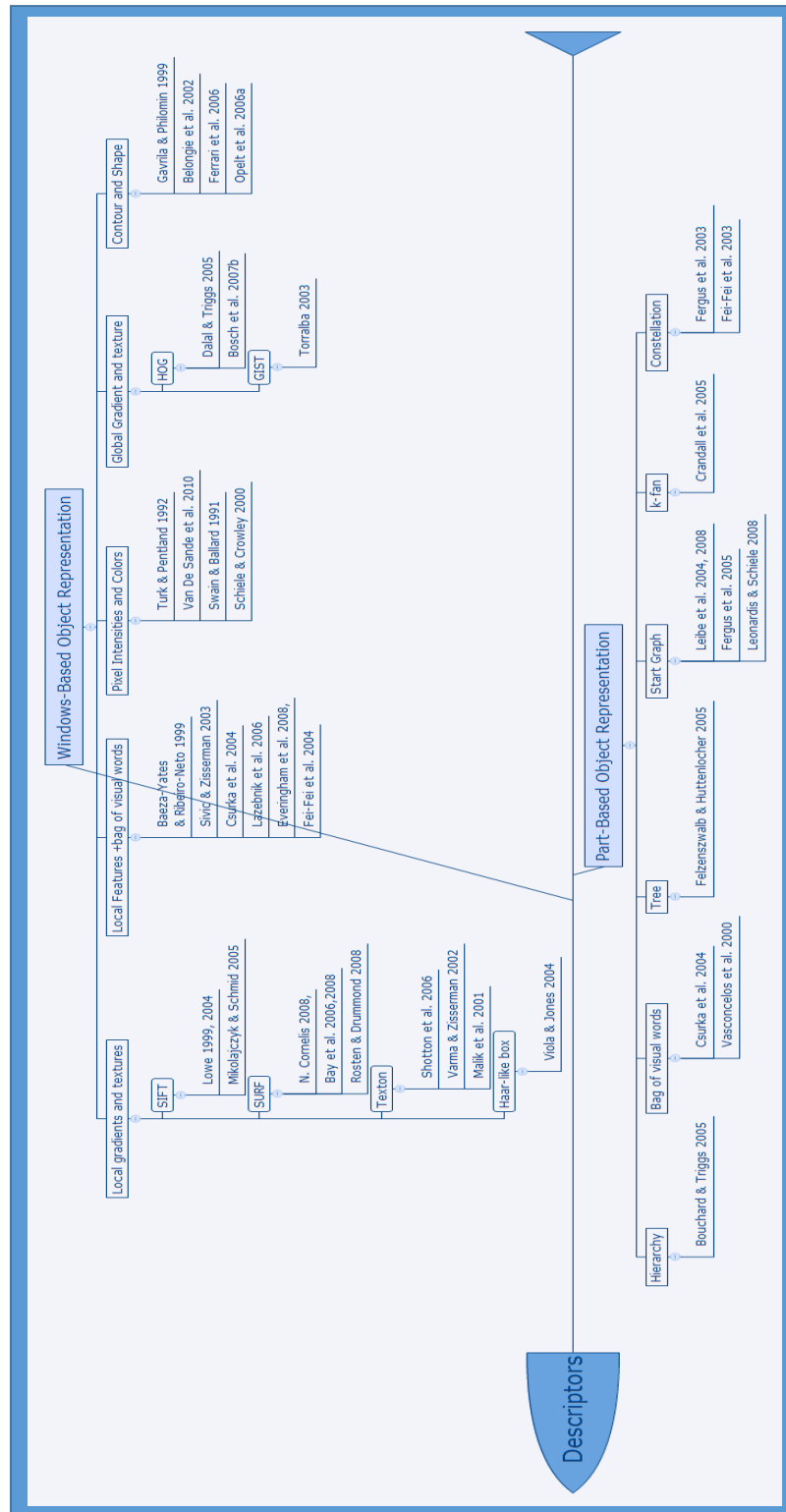


Figure A.11: Object recognition literature divided into windows-based and part-based object representation

APPENDIX B

Traduction française

1 Contexte

La navigation robotique peut se définir comme un processus permettant à un robot mobile de se déplacer de manière autonome d'un point de départ vers une destination finale en utilisant des données sensorielles sur son environnement qui, dans la plupart des cas, peut être qualifié de dynamique et d'imprévisible.

Des robots mobiles, munis de roues ou bipèdes, sont utilisés dans des applications très diverses, dans des contextes militaires, civils ou industriels, à vocation de production, d'aide à la sécurité ou d'assistance. Il existe différentes approches de la navigation, elles ont toutes comme objectif commun de conduire le robot à sa destination finale d'une façon sûre.

Afin de mieux discerner ces approches, ainsi que la motivation de ce travail, les éléments relatifs à la navigation sont présentés ci-dessous à travers un scénario quotidien de navigation.

Imaginez que vous arriviez pour la première fois devant un grand bâtiment inconnu et que vous soyez tenu de trouver votre chemin à l'intérieur de celui-ci jusqu'à une destination finale. Selon le but de votre visite, les contraintes de temps et l'information dont vous disposez sur le bâtiment, plusieurs stratégies peuvent être envisagées. Vous pouvez par exemple utiliser un plan du bâtiment (*navigation à partir d'un plan*). Grâce à celui-ci, vous êtes capable de planifier (*planification hors ligne*) le chemin qui vous conduira à la destination désirée. Il est possible de lire le plan dès l'entrée du bâtiment ou bien de l'imprimer ou encore d'en disposer sous format numérique sur un téléphone ou une tablette. Dans le premier cas, un effort de mémorisation est nécessaire : localisation d'amers susceptibles d'être aperçus dans l'environnement, distance à parcourir entre eux...

Si le plan ou une version à jour de celui-ci n'est pas disponible, il est toujours possible de le construire (*navigation s'appuyant sur la construction d'un plan*) simultanément à votre navigation et à l'apprentissage de la trajectoire vous conduisant (*planification de chemin en ligne*) à votre destination finale.

Dans les cas de *navigation à l'aide d'un plan* ou en *s'appuyant sur la construction d'un plan*, les plans, ou plus généralement les cartes, peuvent disposer d'une métrique (*carte métrique*) ou montrer simplement une suite de repères et de lieux importants localisés par rapport à un point de référence (*carte topologique*).

Une option différente serait de trouver la destination sans utiliser de plan (*navigation sans l'aide d'un plan*) en explorant le bâtiment selon diverses heuristiques. En gardant en mémoire des points de repère, vous construisez ainsi votre propre trajectoire au fur et à mesure de votre avancée (*planification de chemin en ligne*) jusqu'à arriver finalement à votre destination.

Les approches décrites ci-dessus peuvent être utilisées dans des environnements aussi bien intérieur qu'extérieur. Chacun d'eux présente ses avantages et inconvénients.

Par exemple, bénéficier du plan en amont de la navigation vous permet d'atteindre plus rapidement la destination désirée. A tout moment il vous est possible également de connaître

vosre position relativement au plan. Il est cependant nécessaire que ce plan ait été préalablement construit et que l'environnement soit resté identique.

Construire votre propre plan en fonction de ce que vous percevez pendant la navigation vous permet de mettre à jour la base de connaissance sur le bâtiment et de transcrire en temps réel la trajectoire la plus adéquate compte-tenu des éventuelles modifications rencontrées. Toutefois, une construction *online* nécessite temps et efforts, et peut s'avérer complexe à mener si votre propre position n'est pas clairement connue relativement au point de départ ou d'arrivée.

De manière identique à l'approche précédente, naviguer sans carte à l'aide d'heuristique peut conduire à découvrir plusieurs options de chemins possibles, dont certaines permettent de contourner des obstacles rencontrés. Suivant les heuristiques choisies, cette tâche requiert un temps d'exploration plus ou moins conséquent pour découvrir les solutions potentielles. Il est également nécessaire de munir ces heuristiques d'effets mémoire afin de diminuer de manière conséquente le temps d'exploration lors d'un second parcours ou si vous souhaitez préciser à chaque instant le chemin choisi.

Dans le contexte de la navigation autonome de robots mobiles, de nombreux systèmes cherchent à apporter une solution en utilisant une des stratégies décrites ci-dessus. Le processus de navigation implique généralement l'utilisation d'une **représentation spatiale de l'environnement**, de la **localisation** et de la **planification des trajectoires** afin d'**exécuter** l'action appropriée en fonction des informations **perçues** de l'environnement sans aide continue extérieure.

Chaque module fonctionnel intégrant le système est en lui-même un vaste champ de recherche. Les relations entre eux et les moyens d'intégration sont définis par une architecture de contrôle. Dans certains cas, dépendant de l'architecture de commande ou de la stratégie de navigation choisie, un ou plusieurs modules peuvent être omis dans le processus global.

L'architecture de contrôle représente l'élément essentiel dans la définition du système complet de navigation. Elle impose des contraintes fortes sur la façon dont celui-ci peut être contrôlé. Sa complexité dépend, dans une large mesure, des spécificités et des capacités des systèmes robotiques dont les actions doivent s'exécuter la plupart du temps simultanément et de manière asynchrone. La manière dont elle organise, unifie et contrôle les différents modules pour une stratégie de navigation efficace est fortement déterminée et limitée par le **champ applicatif et la mise en œuvre de la plateforme robotique**. Par conséquent, les systèmes robotiques peuvent être classés en fonction de ces deux facteurs.

Selon le point de vue du champ applicatif, différents scénarios auxquels sont soumis les systèmes robotiques tels que les robots industriels, domestiques ou ménagers, de service, militaires ou encore spatiaux, doivent être pris en considération. Par exemple, dans le cas des scénarios déterministes où l'environnement reste identique/stationnaire, la tâche planifiée de navigation peut être grandement simplifiée si une représentation du monde, un plan par exemple, est fournie initialement.

Au contraire des scénarios stochastiques, et donc plus réalistes, tels que ceux rencontrés lors d'opérations de recherche et de sauvetage de victimes de catastrophes, où les systèmes robotiques doivent faire face à des situations imprévues et donc réagir en conséquence. Les réponses ou temps de navigation peuvent alors être très variables selon les difficultés

rencontrées. Les robots se doivent donc d'avoir des capacités décisionnelles pour les tâches qui leur incombent, notamment celles relatives à la navigation.

Du point de vue de la mise en œuvre de la plateforme robotique, les systèmes robotiques (robots munis de roues, robots bipèdes (humanoïdes), robots marins (sous-marins), robots aériens, etc.) ont besoin de contrôler des composants physiques tels que des capteurs, des actionneurs, des processeurs afin d'interagir en temps réel avec leur environnement incertain et souvent dynamique. De plus, les contraintes des systèmes embarqués autonomes exigent que les architectures soient suffisamment flexibles pour permettre le remplacement de composants sur le terrain, et aisément adaptables à différentes plateformes pour des missions variées.

Pour gérer une telle complexité, de nombreuses architectures et composants logiciels et matériels associés ont été proposés. La plupart des architectures de contrôle se sont révélées performantes dans le contexte applicatif pour lequel elles avaient été conçues, et ceci indépendamment de la plateforme. Cependant, actuellement, aucune de ces architectures de contrôle n'excelle sur un large éventail d'applications. Elles échouent généralement lorsqu'un scénario différent se présente et que les caractéristiques de la plateforme ne sont plus en adéquation avec la tâche donnée.

2 Cadre de la thèse

Puisqu'il existe une multitude de situations nécessitant autant de solutions architecturales et de mécanismes associés, une solution optimale serait de concevoir et de mettre en œuvre une architecture de contrôle aussi générique que possible, capable de répondre à toutes sortes de contraintes. Une telle architecture doit être suffisamment souple pour permettre l'intégration de nouveaux composants (matériels et logiciels) sans remettre en question ou modifier ceux déjà existants quelque soit leur niveau dans l'architecture; par conséquent elle doit offrir des mécanismes transparents de communication et d'échange de données ; elle doit apporter aux robots une capacité décisionnelle de haut niveau pour effectuer des actions adéquates tout en étant capable d'affiner et d'adapter ses comportements en fonction des objectifs assignés et des changements soudain de l'environnement.

De plus, étant donné qu'une telle architecture exécute (parfois simultanément) des actions multiples, elle doit être suffisamment robuste pour gérer les priorités, les dysfonctionnements inattendus et la redondance des informations fournies par les sources de capteurs multiples tout en garantissant une performance suffisamment fiable et sans danger.

La liste des propriétés et capacités qu'une telle architecture doit vérifier est cependant infinie, même si nous n'en percevons qu'un nombre limité notamment pour des raisons technologiques, sociétales, éthique ...

Nous pourrions, de manière plus générale encore à l'architecture adaptée dotée de propriétés établies, réfléchir à une architecture adaptative capable de se transformer en fonction des conditions environnementales rencontrées ou de l'évolution des objectifs assignés.

L'auteure de ces travaux de recherche est consciente que réussir à concevoir et à mettre en œuvre une telle architecture générique n'est pas une tâche aisée, et que si celle-ci est possible, beaucoup de temps, d'efforts et de ressources seront nécessaires pour y parvenir. Par conséquent, les travaux présentés dans cette thèse doivent être considérés comme un pas vers cet objectif : définir une architecture de contrôle robuste et efficace pour une navigation autonome de robots mobiles dans un contexte de différents scénarios. Pour ce faire, ce travail se concentre exclusivement sur les contraintes données par plusieurs domaines d'application.

Le principal objectif de cette recherche est de proposer une nouvelle architecture de contrôle robotique capable de s'adapter à différents scénarios de navigation où un robot doit faire face à des situations inattendues. A cet effet, le travail présenté dans cette thèse a été mené dans un contexte de navigation à l'intérieur d'un bâtiment d'un robot humanoïde s'appuyant sur la perception visuelle, via une caméra comme seul capteur de perception.

Deux scénarios très différents (dont beaucoup d'autres peuvent découler) sont ici considérés :

1. Un *scénario déterministe* qui suppose que l'environnement reste toujours identique/stationnaire,
2. Un *scénario stochastique* où le robot doit faire face à des imprévus au cours de sa navigation.

La distinction de ces deux scénarios suggère implicitement qu'ils soient tous les deux définis par rapport à une connaissance préalable de l'environnement et à sa mise en correspondance avec sa perception réelle. Par conséquent lors de la navigation, le robot recherche l'information attendue dans son environnement et la compare avec celle perçue.

1. Ainsi, quand il y a correspondance, le scénario de navigation est dit *déterministe*.
2. Dans le cas contraire, le *scénario* est supposé être *stochastique*. Le robot doit rechercher par lui-même son chemin pour arriver à destination grâce à des algorithmes et stratégies de navigation ne nécessitant aucune connaissance *a priori*.

Depuis sa création jusqu'à son développement et son achèvement, ce travail a reposé sur deux piliers importants : les stratégies de navigation s'appuyant sur la perception visuelle et les différents paradigmes sur lesquels reposent les architectures de contrôle.

3 Etat de l'art

L'état de l'art des stratégies de navigation reposant sur la vision a été présenté dans le chapitre 2 du manuscrit en suivant une structure transversale proposée par l'auteure. Il est décrit selon différents modules fonctionnels (**perception visuelle, modélisation de l'environnement, localisation et planification des trajectoires**), dont leur implication dans la tâche de

navigation dépend fortement de deux types de stratégies de navigation : **la navigation sans l'aide d'un plan** et **la navigation avec plan**. Ce dernier comprend aussi la **navigation s'appuyant sur la construction d'un plan**.

Navigation à l'aide d'un plan : cette approche consiste à fournir au robot un modèle de l'environnement sur lequel il peut s'appuyer lors de sa navigation ou qu'il pourra enrichir via ses capteurs pendant la navigation.

Navigation avec plan : l'utilisation d'une connaissance *a priori* de l'environnement facilite grandement le processus de navigation. Le robot peut se localiser lui-même dans l'environnement en estimant sa position actuelle par rapport à un référentiel, planifier une trajectoire selon les informations extraites, et enfin se déplacer pour atteindre sa destination finale. C'est pourquoi on appelle également cette approche *navigation avec carte*.

Navigation s'appuyant sur la construction du plan : la construction d'un plan fait référence au processus de création de modèles 2D ou 3D, géométrique ou topologique, via l'utilisation de capteurs pendant la navigation. Deux approches peuvent être distinguées : les systèmes qui construisent un plan en amont et l'utilisent pour la localisation du robot (construction hors ligne) et les systèmes qui construisent la carte en ligne et permettent une localisation du robot simultanément dans l'environnement (SLAM).

Navigation sans l'aide d'un plan : cette stratégie de navigation consiste à réaliser une navigation autonome, sans utiliser, ni créer un modèle de l'environnement, ni avant, ni durant la navigation. Par conséquent, afin de naviguer et se localiser, le robot doit appréhender visuellement des repères saillants (amers) afin de le guider dans ses déplacements : murs, portes, angles, etc.

Le robot se questionne sur l'environnement, observe la scène, extrait les caractéristiques les plus pertinentes des amers, sauvegarde leur position et se localise grâce à eux.

Puisque les approches algorithmiques utilisées dépendent principalement des capteurs ou du type d'amer, nous les distinguons en conséquence. Nous souhaitons souligner que la plupart de ces techniques sont également utilisées dans le contexte cartographique avec cependant un objectif de mise en correspondance.

Ces types de navigation intègrent une combinaison de modules fonctionnels interconnectés :

Perception : la perception fournit l'information d'entrée du module de contrôle. Il s'agit du processus d'interprétation et de transformation de l'information sensorielle du robot avec son environnement et d'autres entités externes vers une représentation qui peut être utilisée pour un traitement ultérieur ou pour d'autres actions.

Modélisation de l'environnement : grâce à l'utilisation de différents capteurs, il est possible de représenter l'environnement de navigation par un plan et d'utiliser celui-ci directement pour calculer le trajet afin d'exécuter la tâche de navigation. Il est également possible de le reconsidérer *a posteriori* pour mettre à jour de nouvelles informations afin d'avoir une meilleure précision dans sa construction, et ainsi, obtenir une localisation plus précise. Une

telle représentation cartographique peut être construite en amont ou lorsque le robot découvre son environnement. Dans les deux cas, la représentation peut être scindée selon deux catégories via des cartes métriques ou topologiques.

Localisation : cette propriété témoigne de la capacité du robot à établir sa propre position et son orientation dans l'environnement. Les techniques de localisation nécessitent de disposer de l'information sur l'environnement (point initial ou de destination, plan plus ou moins synthétique, ...). Plusieurs techniques ont été proposées. Leurs principales différences dépendent de la nature des capteurs du robot, de l'environnement et de l'information initiale disponible.

Planification de trajectoires : la planification de trajectoires est une extension de la tâche de localisation. Elle nécessite de déterminer dans le même référentiel la position initiale et finale que le robot doit atteindre, afin d'être en mesure de planifier un chemin optimal qui ne présentera pas de collision.

Le critère de performance optimale dépend des objectifs de l'application. Il peut être choisi en termes de distance (chemin le plus court), de temps (le plus rapide) ou d'énergie (consommation d'énergie la plus faible).

Les algorithmes de planification de trajectoires sont évalués selon leur complexité de calcul. Les approches peuvent être scindées en deux catégories selon la disponibilité ou l'absence d'une représentation complète de l'environnement.

Finalement, le déplacement du robot selon la trajectoire planifiée est contrôlé en permanence en jouant sur ces actionneurs afin de s'adapter aux changements de l'environnement comme l'apparition d'obstacles.

Les architectures de contrôle des robots définissent comment ces capacités peuvent être intégrées et mises en interaction pour construire et développer une navigation autonome. Pour cette raison, la complexité de la navigation d'un robot mobile dépend principalement du type d'architecture de contrôle choisi. En conséquence, de multiples architectures ont été proposées dans la littérature pour concevoir et développer des systèmes de commande robustes, flexibles, fiables et performants. Chacune de ces architectures de contrôle sous-tendent de nouveaux concepts.

Dans le contexte de ce travail, l'état de l'art des différents paradigmes a été présenté au chapitre 3 à partir de deux points de vue.

Selon le **point de vue conceptuel**, le problème de la navigation robotisée peut être résolu en traitant les informations soit de façon ascendante, soit de façon descendante.

Même si l'approche bottom-up est beaucoup plus récente, elle prend une place aussi importante que l'approche top-down. Alors que cette dernière permet de décomposer le problème en commandes de bas niveau pour aider le robot à planifier ses mouvements, les modèles bottom-up sont appropriés pour naviguer dans des environnements inconnus et dynamiques. L'utilisation conjointe permet au robot d'apprendre à faire face à des situations et des difficultés imprévues. Les modèles bottom-up sont en effet particulièrement adaptés aux changements et ne nécessitent pas une énorme complexité informatique contrairement aux modèles top-down.

Cependant, chacun d'eux présentent des lacunes, intrinsèques à leur conception, qui doivent encore être surmontées. En raison du traitement séquentiel du processus de l'information, le

processus de navigation dans les modèles top-down peut présenter des délais importants de réaction. De plus, la quantité d'espace nécessaire pour sauvegarder toutes les connaissances préprogrammées peut dépasser les capacités mémoire du robot. En outre, le dysfonctionnement de l'un des modules peut provoquer la défaillance de l'ensemble du système. De même, avec une approche bottom-up il est très difficile d'atteindre une complexité de niveau supérieur. Le temps requis pour apprendre une tâche ou pour faire émerger une certaine intelligence du comportement peut être un frein.

Une solution est de combiner à la fois les approches top-down et bottom-up de manière à ce que l'architecture de contrôle puisse d'une part bénéficier d'une connaissance préprogrammée de l'environnement, et d'autre part, s'adapter aux environnements du monde réel grâce à l'émergence de comportements résultant de l'interaction avec l'environnement.

Selon le **point de vue fonctionnel**, quatre paradigmes peuvent être distingués :

- **l'approche délibérative** s'appuyant sur une connaissance préalable et l'observation interne des actions ou des états ;
- **l'approche réactive** reposant sur un modèle stimulus-réponse apportant la possibilité au robot de faire face à des environnements très dynamiques et imprévisibles ;
- **l'approche hybride**, qui combine les avantages des approches réactives et délibératives tout en diminuant leurs inconvénients ;
- Enfin, **l'approche comportementale** composée d'une collection de comportements indépendants.

Chacune des approches présentées est construite dans le but de permettre aux robots d'effectuer de façon autonome une variété de tâches dans différents domaines applicatifs.

Une approche peut donner entière satisfaction pour l'exécution d'une tâche donnée, mais peut échouer pour une autre tâche où un objectif différent est requis. Par conséquent, le choix d'une approche d'architecture de contrôle dépend principalement des propriétés du problème, du type de tâche souhaitée, de l'optimalité requise et des informations disponibles. De plus, elle est étroitement liée aux contraintes matérielles et logicielles du robot.

Par exemple, **les systèmes délibératifs** fournissent un raisonnement et une planification optimaux si la représentation de l'environnement s'avère exacte. Cela implique que l'environnement soit statique. Par conséquent, ces systèmes sont très bien adaptés pour des environnements structurés et fortement prévisibles, en particulier dans les domaines où le robot effectue périodiquement une tâche donnée. Ces systèmes ne conviennent pas en robotique située.

Les systèmes réactifs donnent de très bons résultats sur des environnements dynamiques pour lesquels une réponse et une réaction immédiates sont essentielles (e.g. évitement d'obstacles). Cependant, le fait qu'il ne dispose pas d'une représentation du monde et de connaissances des actions passées et futures pose problème lorsque la planification, l'apprentissage ou la sauvegarde en mémoire sont nécessaires.

Les systèmes hybrides préservent les avantages des approches précédentes tout en palliant à leurs inconvénients. Ils sont adaptés aux environnements nécessitant des modèles internes et une planification à long terme dont les besoins d'exécution temps réel ne seraient pas contraints par les couches supérieures.

Finalement, il est possible de conclure que **les systèmes comportementaux** comprennent quasiment tous les avantages des trois autres approches, tout en étant capable d'apprendre et de s'adapter facilement à des environnements changeants de manière significative. De plus, leurs composants (comportements) et leurs interconnexions permettent au système de planifier, d'éviter les erreurs passées et d'utiliser une représentation active si nécessaire. Les comportements sont conçus à partir d'une variété de niveaux d'abstraction, facilitant la construction ascendante des systèmes s'appuyant sur le comportement. Cependant, la difficulté de mettre en œuvre une telle architecture peut représenter un inconvénient majeur.

Même si l'architecture de contrôle idéale n'a pas encore été développée, plusieurs travaux de recherches tentent d'améliorer et d'en proposer de nouvelles en combinant, le plus souvent, le meilleur des approches, tels que, par exemple, le raisonnement et la planification optimale, et la capacité de répondre rapidement aux changements dynamiques du milieu. La combinaison de systèmes s'appuyant par essence sur le comportement avec ceux construits autour d'une représentation globale du monde peut être une bonne solution. Par exemple, l'architecture (AuRA) facilite la planification et le raisonnement en utilisant directement un planificateur pour sélectionner les comportements \ cite {arkin1997aura}. De même, l'architecture hybride à trois niveaux 3T utilise des comportements dans sa couche réactive \ cite {bonasso1995experiences}.

4 Solution proposée

Afin de répondre aux objectifs présentés précédemment et de prendre en compte les spécificités données par les différents scénarios décrits, nous proposons l'architecture de contrôle RHIZOME (Robotic Hybrid Indoor-Zone Operationa ModulE en anglais).

L'architecture est composée d'un réseau de neurones artificiels composé de nœuds interconnectés qui calculent des valeurs de sortie à partir d'une ou plusieurs entrées reçues, et ceci de manière similaire à un réseau neuronal biologique présent dans le système nerveux d'un organisme. Ces modèles de calcul ont des propriétés telles que la mémorisation associative, l'apprentissage et le traitement parallèle d'informations multiples. Leur interconnectivité est telle que tout le système est capable de s'auto-adapter en fonction de ses entrées résultantes de l'interaction entre le robot et l'environnement, développant ainsi de nouveaux comportements.

Lors de la conception et la mise en œuvre de l'architecture RHIZOME, un certain nombre de problématiques seront abordés dans le but de positionner les solutions existantes. On trouvera ci-après, selon un traitement séquentiel, les questions qui nous paraissent les plus pertinentes, puis une description des caractéristiques et du fonctionnement de l'architecture RHIZOME.

4.1 Problèmes abordés et contributions

La conception et la mise en œuvre d'une architecture de contrôle capable de s'adapter à différents scénarios nécessite de satisfaire des exigences très différentes. A cet effet, les problèmes abordés dans cette thèse ont été motivés par les contraintes des scénarios décrits précédemment (déterministe et stochastique). Une approche cohérente que nous tentons de

justifier ci-dessous a été adoptée. Elle fait état d'une liste de questions ou de points de décision et consolide progressivement l'espace des solutions sous la forme de l'architecture RHIZOME.

1. Comment l'environnement de navigation peut-il être utilisé au mieux afin d'aider la navigation du robot ?

Si l'on reprend l'exemple décrit précédemment tiré d'un scénario de la vie quotidienne pour lequel trois stratégies de navigation différentes ont été envisagées pour atteindre une destination finale, il convient de noter qu'il existe un dénominateur commun entre ceux-ci: l'utilisation comme points de référence de **repères saillants**, le plus souvent stationnaires.

Certes, lorsque vous utilisez un plan, l'information (après avoir planifié le chemin complet) peut être compressée d'une manière ou d'une autre. Cependant, la compression reste limitée par le fait qu'il est nécessaire de s'assurer en permanence des éventuels changements vis-à-vis du plan, et donc de garder une représentation suffisamment complète des connaissances/modèles conjointement aux amers.

De même, lorsqu'aucune représentation du monde n'est pas disponible, les amers jouent un rôle important dans la tâche de navigation : en navigation maritime, avant l'arrivée des techniques modernes de navigation, les étoiles polaires servaient de points de référence car elles ne disparaissaient pas à l'horizon ; de même, dans notre vie quotidienne, des repères (désignés dorénavant par « symboles de navigation ») nous aident dans nos déplacements ; les panneaux de signalétique tels que ceux trouvés sur les routes, dans les stations de métro et aéroports, ceux de sécurité incendie dans les bâtiments, nous aident d'une manière ou d'une autre à nous guider vers nos destinations.

Deux types de symboles de navigation sont donc considérés dans ce travail: les **symboles artificiels** et les **symboles dits « naturels »**. Les **symboles artificiels de navigation** se réfèrent à des symboles prédéfinis préalablement placés le long des parcours de navigation. Inversement, les **symboles naturels de navigation** se réfèrent à des motifs naturels saillants (i.e., amers) suffisamment remarquables pour être considérés comme des points de référence.

L'utilisation de ces deux types de symboles peut être envisagée selon les points de vue de deux paradigmes. A l'origine le *paradigme symbolique* de l'intelligence artificielle supposait que la manipulation des symboles était suffisante pour traiter de nombreux aspects de l'intelligence tel que celui du contrôle d'actions même complexes dans les machines. Cela s'est avéré pertinent lorsque la sortie d'une machine manipulant des symboles s'appuie sur des entrées et des règles de décision bien définies (e.g. systèmes experts, architectures de commande robotique délibérative, etc.).

Il s'agit d'un paradigme qui a prédominé durant trois décennies. Cependant, le manque de robustesse des systèmes de contrôle en présence d'incertitude a conduit des travaux de recherche à s'intéresser à l'approche sous-symbolique : le paradigme de la *nouvelle IA*. Contrairement au paradigme symbolique, le paradigme de la nouvelle IA n'utilise pas une représentation spécifique du monde pour intégrer l'intelligence, mais pose plutôt

l'hypothèse que l'intelligence émerge de comportements simples résultant de l'interaction du robot avec son environnement immédiat via des liens sensori-moteurs. Plutôt que d'utiliser un seul planificateur centralisé, le système intègre la connaissance de manière répartie sur plusieurs unités de type comportementale, l'ensemble global constitue donc un réseau d'unités interconnectées.

Par conséquent, tandis que les **symboles artificiels de navigation** sont utilisés dans ce travail comme des symboles de haut niveau pouvant être manipulés par le robot dans un scénario déterministe, les **symboles naturels de navigation** perçus de son environnement par le robot permettent de pallier l'absence de symboles artificiels dans le contexte des scénarios stochastique. Ce choix est justifié ci-dessous à travers un jeu de questions-réponses.

2. Comment le robot peut-il accéder à sa destination finale de manière la plus efficace et la plus simple ?

Le scénario le plus simple est celui dans lequel la représentation de l'environnement est calculée à l'avance et celui-ci reste inchangée pendant la navigation. Une telle configuration implique un scénario déterministe. La représentation de l'environnement est alors présentée au robot avant le début de l'activité de navigation.

Donc, **s'appuyer sur un plan** pour effectuer une tâche de navigation nous semble une solution optimale, la représentation du monde étant fournie à l'avance.

Du point de vue architectural, une **architecture délibérative** et un traitement d'information de type **top-down** semblent être les plus adaptés pour traiter les informations extraites du plan et pour planifier le parcours. Les architectures délibératives ont représenté le paradigme dominant pendant des années pour la construction de robots. Elles s'appuient comme nous l'avons vu sur un modèle donné au préalable et traitent l'information de manière descendante. Dans le cas de notre étude, le modèle correspond à l'environnement dans lequel le robot doit naviguer.

• Mais quel type de plan ?

Les plans de bâtiments (tels que les hôpitaux, les écoles, les complexes résidentiels ou les usines) ne sont généralement pas aussi facilement disponibles sous format numérique que les plans de villes, de pays entiers, etc. Il s'avère souvent que face à un bâtiment inconnu, nous soyons obligé de faire appel aux plans sur support matériel disponibles à l'entrée du bâtiment ou se présentant sous forme papier.

Un tel plan de bâtiment représente un moyen des plus rapides pour accéder à une information complète sur l'intérieur d'un bâtiment. Nous utilisons dans nos travaux de recherche une telle modalité pour représenter l'information sur l'environnement de navigation. Ce plan se présentera de manière très synthétique selon un ensemble de symboles artificiels dédiés à la navigation.

L'objectif sera d'extraire une séquence de ces symboles relativement à la trajectoire souhaitée. L'expression synthétique du plan doit simplement porter l'information topologique afin de permettre cette extraction.

Lorsque seule la séquence de symboles est l'information disponible, nous nous heurtons au *symbol grounding problem*. Celui-ci fait référence au sens porté par chaque symbole (e.g., la signification directionnel du symbole, dépendant ou non du contexte, son interprétabilité sans autre information que le symbole lui-même...), et nécessite une interaction constante entre le robot et l'environnement pour sa résolution.

Dans le cas où le robot a la possibilité et les capacités de lire et d'interpréter lui-même le plan, il peut non seulement déterminer la séquence de symboles afin de planifier son parcours mais également leur signification ainsi que la distance inter-symboles. À cette fin, la métrique habituellement indiquée dans certains plans peut être utilisée.

Indépendamment de l'information obtenue (séquence de symboles uniquement ou séquence de symboles et leur signification), l'interaction avec l'environnement demeure essentielle, le robot ayant besoin de comparer l'information obtenue avec celle perçue dans l'environnement réel. Le robot doit non seulement être en mesure de corroborer les informations fournies par le plan (*reconnaissance des symboles*), mais aussi connaître l'action à effectuer parmi plusieurs autres actions possibles résultant de la reconnaissance ou non du symbole.

Interagir avec l'environnement reste une tâche difficile, car celui-ci est la plupart du temps dynamique et imprévisible ; et même si l'utilisation d'une information *a priori* permet d'alléger la tâche de navigation, un bon système doit être suffisamment souple pour faire face à tout changement imprévu. Par conséquent, il est nécessaire de rechercher une solution permettant de gérer ce problème d'incertitude et donc d'exécution d'une action donnée parmi beaucoup d'autres, ce qui nous amène à la question suivante.

3. Quel type de mécanisme ou modèle semble être bien adapté pour gérer les environnements dynamiques ?

Afin d'aborder la problématique de navigation dans un environnement dynamique, nous choisissons de nous appuyer sur un modèle bio-inspiré. En effet, les humains ont une capacité incroyable, grâce notamment à leur capacité d'apprentissage, à s'adapter aux changements imprévus en réagissant en conséquence. Cette adaptabilité est due aux capacités de leur cerveau et de ses composantes.

Le système neuronal a donc fait l'objet de nombreuses études et a inspiré plusieurs modèles dans le domaine de l'intelligence artificielle. L'architecture que nous proposons s'appuie sur ces différents modèles.

Les modèles neuronaux du *connexionnisme* précédemment connu sous le nom de *traitement distribué parallèle* (Parallel distributed processing en anglais) ou de modèles PDP, utilisent des unités de traitement simples et souvent uniformes pour traiter l'information. La mémoire est portée localement par l'interaction d'un grand nombre de ces unités via des signaux stimulants et inhibiteurs. Chaque unité reçoit une valeur d'entrée émanant des unités voisines, exécute une fonction selon les entrées reçues et calcule une valeur de sortie. La configuration intrinsèquement distribuée permet de réaliser simultanément le calcul de plusieurs unités, ce qui permet d'accélérer le traitement de l'information.

La représentation des connaissances dans les modèles PDP n'est pas présente dans un état ou une mémoire à long terme comme dans d'autres modèles conventionnels. Au contraire, la connaissance fait partie du processus lui-même et détermine le déroulement de celui-ci dans le sens où la mémoire à long terme est sauvegardée dans les liens de renforcement des unités, tandis que la mémoire à court terme est enregistrée dans les états des unités. Les unités peuvent représenter différentes structures selon le modèle. Par exemple, une unité simple peut exprimer une caractéristique, un symbole ou un concept. Elle peut également symboliser des éléments abstraits qui, en s'assemblant avec de nombreux autres, peuvent représenter une entité entière ou un concept.

Une propriété extrêmement importante de ces modèles provient du fait qu'il est possible d'apprendre par l'expérience en utilisant un mécanisme de modulation permettant d'ajuster la connexion entre les unités. Il existe différentes règles pour ajuster les connexions. La plupart d'entre elles dérive de la règle d'apprentissage proposée par Hebb (1949) qui stipule que lorsque deux unités sont stimulées simultanément, la connexion entre elles est renforcée.

Du point de vue architectural, les architectures s'appuyant sur le comportement se composent d'une collection de modules comportementaux organisés de façon distributive et parallèle, comme les modèles PDP. Ils sont généralement exécutés simultanément et de façon asynchrone et, en les rassemblant dans des environnements complexes, des comportements émergents peuvent se produire. Puisqu'il n'y a pas de contrôle central parmi les modules comportementaux, toutes les couches sont interconnectées permettant ainsi une communication interne décidant de la meilleure action ou du meilleur comportement à effectuer.

Par conséquent, **une architecture comportementale** selon un traitement **top-down** semble être une bonne option pour prendre en compte l'incertitude des interactions en temps réel dans des environnements dynamiques et imprévisibles. A ce sujet, l'architecture de *subsumption* de Brooks s'est montrée particulièrement efficace pour surmonter ce type de problème.

Notre travail s'inspire donc des architectures comportementales pour une plus grande robustesse aux changements. Plus particulièrement, le mécanisme, les propriétés et les composants de l'architecture que nous proposons s'appuient sur l'architecture **PerAc** inspirée principalement des travaux de Brooks, Edelman and Grosseberg, et proposée par Gaussier et Zheren comme structure neuronale organisée. L'architecture **PerAc** n'utilise pas une représentation du monde pour contrôler l'action du robot. Au contraire, elle suit un mécanisme de perception-action qui évolue constamment grâce à l'interaction dynamique entre le robot et son environnement (voir chapitre 3 pour plus d'informations).

- **Quelle action doit être effectuée lorsque l'information attendue n'est pas perçue dans l'environnement de navigation ?**

L'absence de *symboles artificiels attendus* de navigation dans un environnement stochastique oblige le robot à opter pour une stratégie différente forçant celui-ci à rechercher de nouveaux points de référence pour se repérer. Pour ce faire, la recherche de

symboles naturels de navigation présents dans l'environnement (telle qu'elle est présentée dans les stratégies de navigation sans l'aide de plan) semble être une bonne solution.

Cependant, notre travail va au-delà de la détection de symboles naturels de navigation, il s'inspire en effet de l'approche robuste proposée par [Gaussier 2002] qui permet la reconnaissance de lieux.

En effet, un lieu peut être identifié comme un point de référence stable qui peut être appris en mémorisant l'emplacement de formes saillantes perçues dans le champ visuel panoramique du robot. La problématique de reconnaissance d'un lieu en cours de navigation revient à reconnaître, aux mêmes emplacements, le même ensemble de formes apprises.

La robustesse d'une telle approche réside dans le fait que même si une ou plusieurs formes caractérisant l'endroit sont supprimées ou ne sont pas visibles, l'endroit peut encore être reconnu. De plus, grâce à un procédé de triangulation s'appuyant sur ces motifs, il est possible d'obtenir des informations sur la position du robot dans ce lieu.

L'architecture de contrôle doit en permanence comparer information perçue et connaissances afin de décider de l'action à exécuter. Comment cette comparaison est finalement réalisée, c'est l'objet de la dernière question.

4. Comment prendre en compte simultanément les deux sources d'information et agir en conséquence?

Comment combiner en temps réel l'information a priori de l'environnement, connue en amont, et l'information dynamique perçue ? Comment combiner architecture délibérative et comportementale ? De manière plus générale, comment combiner plusieurs stratégies de navigation ?

De manière plus synthétique, comment un modèle connexionniste peut-il être associé à un modèle symbolique ?

Pendant longtemps, ces deux courants de l'intelligence artificielle ont été considérés comme opposés l'un à l'autre. Tandis que pour la première, la connaissance est présente dans les liens de renforcement des unités de réseau de l'approche connexionniste, cette même connaissance, dans la seconde approche, est représentée par les chaînes de symboles.

Malgré cette différence majeure, certains connexionnistes conviennent qu'il est possible de réunir les deux paradigmes en une seule architecture connexionniste. Ils postulent qu'il devrait être possible d'implémenter un traitement symbolique dans un réseau de neurones compte tenu de la capacité des humains à effectuer des tâches de manipulation de symboles de haut niveau en dépit de la configuration neuronale du cerveau.

Par conséquent, suivant cette même ligne de pensée, l'architecture proposée dans ce travail profite des propriétés caractérisant les réseaux neuronaux pour fusionner les deux informations dans une structure neuronale.

On a coutume de distinguer sur les architectures neuronales classiques, tels que les réseaux neuronaux récurrents (RNN) ou les réseaux neuronaux Feed-Forward (FNN), trois types d'unités neuronales différentes : la couche d'entrée, la couche de sortie et la couche cachée. La manière selon laquelle notre architecture relie les unités et les groupes d'unités est différente de celle des réseaux classiques, ce qui la démarque profondément de ces derniers. Par conséquent, nous introduisons un autre type d'unité, celui de « couche interne ». Le lecteur peut se référer au chapitre 4, qui explique en détail les connexions entre unités.

En ce qui concerne le problème de la sélection d'actions, les propriétés des neurones artificiels apportent une solution tout à fait pertinente et inhérente à leur structure. Dans son livre, *The Mindful Brain*, Edelman (1978) développe sa théorie du darwinisme neuronal, où il évoque la plasticité des réseaux de neurones vis-à-vis de l'environnement. L'interconnexion entre les neurones est renforcée par l'expérience : lorsqu'un stimulus externe ou interne est reçu par le système, différents neurones sont simultanément activés et transmettent l'information aux neurones voisins. La sortie est alors la résultante des différentes activations stimulées par une source d'entrée donnée.

Les réseaux neuronaux artificiels ont la capacité de modéliser n'importe quelle fonction donnée. Par conséquent, il est possible de définir différentes fonctions d'activation tout au long du réseau afin de déclencher des comportements différents. En conséquence, tout le système fonctionne en parallèle et un « mécanisme concurrentiel » permet de décider du meilleur comportement ou action à effectuer pour contrôler le robot en fonction du stimulus reçu.

Le fait d'analyser, de réorganiser et de synthétiser les solutions présentées ci-dessus a conduit l'auteure à comprendre que la conception d'une architecture de contrôle capable de répondre à des contraintes de scénarios différents n'est possible qu'en conciliant les différences entre les paradigmes jusqu'ici proposés. Ainsi, au lieu d'aborder la problématique en suivant un seul paradigme ou selon un unique chemin de pensée, il est possible de créer une synergie en combinant plusieurs approches dans une structure transversale : l'architecture RHIZOME.

4.2 L'Architecture RHIZOME

L'architecture RHIZOME a émergé par la volonté de fournir une autonomie suffisante aux robots mobiles pour leur permettre, lors de leur navigation dans l'environnement, de s'adapter aux situations imprévues. Elle se compose d'une architecture hybride comportementale qui fusionne dans une structure neuronale transverse information *a priori* et information visuelle perçue en temps réel dans l'environnement.

L'information *a priori* de l'environnement est seulement utilisée ici pour confirmer l'information visuelle, contrairement à la plupart des architectures hybrides qui l'utilisent pour contrôler directement les actions du robot. En outre, au lieu d'utiliser une trajectoire complète, l'architecture utilise, sous la forme d'une séquence codant le chemin de navigation, des symboles artificiels. Le robot est muni d'un mécanisme capable d'extraire les deux sources d'information, de les comparer en temps réel et de réagir en conséquence. Lorsque les

symboles de navigation ne sont pas présents ou détectés dans l'environnement, l'architecture de contrôle permet au robot d'apprendre et de reconnaître des lieux à partir de formes saillantes, appelées symboles naturels de navigation, perçues dans l'environnement.

Ainsi, grâce à ces points de repère, artificiels ou naturels, le robot est en mesure d'atteindre sa destination finale et surmonter des situations imprévues lors de la planification.

Comme le présente la figure 1, l'architecture est composée d'une structure comportementale hybride qui combine un module délibératif et un ou plusieurs modules comportementaux.

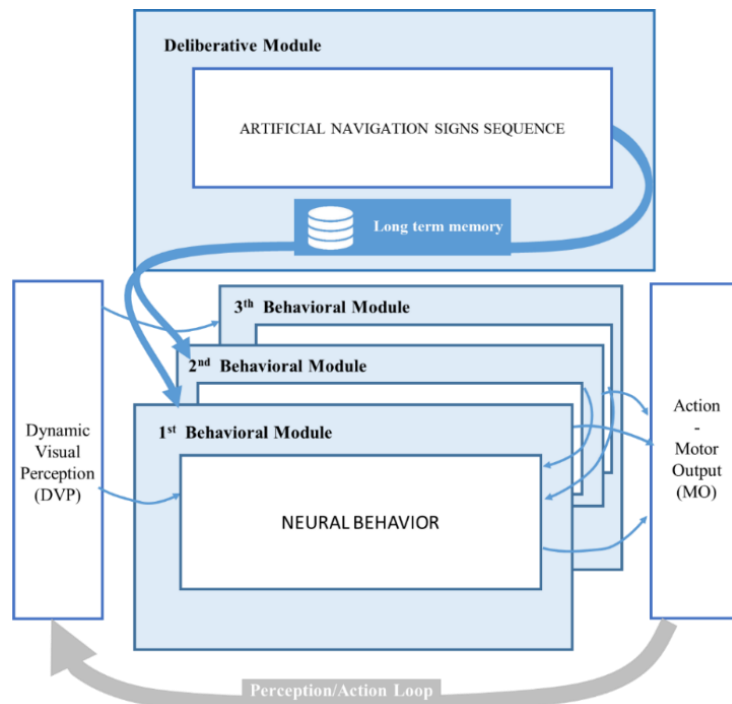


Figure 1 Architecture RHIZOME

Le **module délibératif**, représenté par la partie supérieure de l'architecture (cf. Figure 1) intègre la connaissance *a priori* de l'environnement de navigation. Dans ce travail, cette connaissance se présente sous la forme de symboles planifiant le chemin de navigation. La séquence est intégrée dans les modules comportementaux de l'architecture, soit directement grâce à un programme de commande, soit à partir d'une chaîne de traitement d'images qui l'extraît à partir du plan représentant le bâtiment.

Les **modules comportementaux** représentés par les parties inférieures de l'architecture (cf. Figure 1) s'appuient sur l'architecture PerAc (Perception-Action) et des flux de perception et d'action sous-jacents. Le premier niveau, représenté sur la Figure 2 par la **couche de comportement Réflexe**, utilise un mécanisme réflexe qui contrôle directement les actions du robot à partir des informations extraites de l'entrée perçue. Le deuxième niveau, représenté par la **couche de Reconnaissance**, propose un mécanisme cognitif permettant la reconnaissance en intégrant le flux perceptif précédent et en apprenant les associations

sensori-motrices. La Figure 2 montre plus précisément l'architecture PerAc (gauche) qui est utilisée dans le module comportemental de l'architecture Rhizome (à droite).

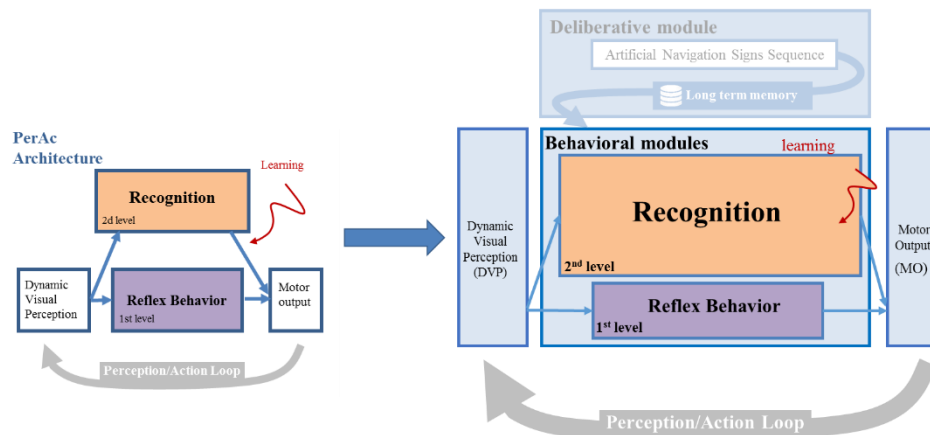


Figure 2. Architecture PerAc (gauche) utilisé dans le module comportemental de l'architecture Rhizome (droite)

L'ensemble du système fonctionne en parallèle et un «mécanisme compétitif» permet de décider du meilleur contrôle, (parmi les différentes couches) fonction du stimulus reçu. Cela est rendu possible grâce aux connexions excitatrices ou inhibitrices, permettant ou empêchant, l'activation des neurones. En outre, lorsque l'apprentissage est nécessaire, une connexion de type modulation, conditionnée par un signal de renforcement, est utilisée.

L'architecture globale suit un cycle perception-action qui signifie que pour chaque information d'entrée provenant de la perception visuelle, il y a toujours une action qui s'exécute, modifiant elle-même la perception de l'environnement pour un nouveau cycle de processus.

Dans le contexte de ces travaux de recherche, l'architecture RHIZOME a été conçue, construite et implémentés à travers trois scénarios pour lesquels trois architectures interdépendantes ont émergé, chacune répondant aux contraintes spécifiques d'un scénario donné.

Rhizome 1 : Scénario Déterministe - Explorer l'environnement avec peu d'information

Rhizome 2 : Scénario Déterministe - Naviguer de manière autonome à partir d'un plan de bâtiment

Rhizome 3 : Scénario Stochastique - Apprendre et s'adapter en fonction des changements imprévus dans l'environnement

Les architectures ne doivent pas être considérées selon une hiérarchie ou selon un modèle d'évolution ascendant. La première architecture a émergé à partir du scénario le plus simple imaginé.

La seconde s'appuie sur les composants de la première, et parce que celle-ci permet d'intégrer de nouveaux composants, fait émerger de nouvelles fonctionnalités répondant ainsi à la problématique de second scénario. Il en va de même avec la troisième architecture. Chaque architecture s'avère donc essentielle.

Rhizome 1 : Explorer l'environnement avec peu d'information

Rhizome 1 est composé des modules nécessaires pour que le robot puisse atteindre sa destination finale dans un contexte de scénario déterministe tout en ayant peu de connaissances sur le monde. Cette dernière, *a priori et* globale, est représentée par une séquence de symboles de navigation qui est définie dans le module délibératif de l'architecture par un programme de commande (voir Module délibératif de la figure 3). Rhizome 1 intègre la séquence de symboles dans un module comportemental afin de permettre au robot de détecter et de reconnaître durant la navigation chaque symbole attendu. De plus, puisque la signification du symbole (sa direction associée) est inconnue, Rhizome 1 permet au robot de la déduire et de l'apprendre en suivant un modèle de stimulus-réponse. Le robot est alors capable d'effectuer, pour des situations similaires à celles rencontrés, les mouvements appris. En conséquence, le module comportemental de l'architecture Rhizome 1 diffère de l'architecture classique PerAc par le fait qu'elle possède un module PerAc imbriqué dans son second niveau. Par conséquent, l'architecture est composée de trois couches comme cela est illustré par la figure 3 dans le module comportemental.

Ci-dessous, nous explicitons chaque couche en parcourant le module comportemental du haut vers le bas de la figure 3.

La première couche du système (SRMA) est en charge de fusionner les informations provenant de la mémoire à long terme et de la perception visuelle pour la phase de reconnaissance des symboles attendus dans l'environnement de navigation. En outre, elle est en charge d'apprendre l'association entre chaque symbole et son mouvement associé. Les deux autres couches utilisent un mécanisme réflexe qui contrôle directement l'action du robot à partir de l'information perçue de l'environnement. La deuxième couche (DDRB) détermine la direction que doit prendre le robot (gauche ou droite) en recherchant l'emplacement du

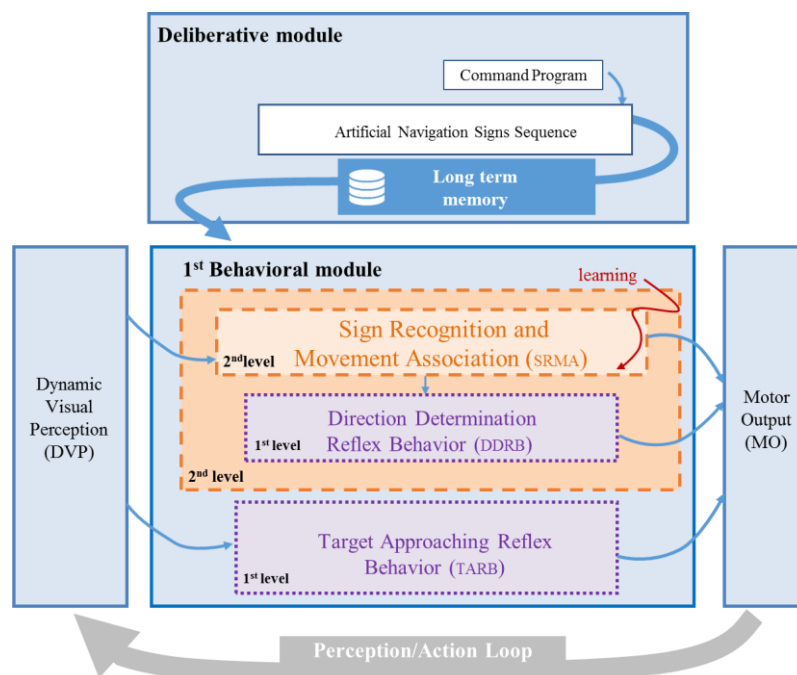


Figure 3 RHIZOME 1 composé d'un module délibératif simple et d'un module comportementale composé lui-même de trois couches.

prochain symbole attendu (dédduit de la séquence), tandis que la troisième couche (TARB) permet au robot de se rapprocher du symbole lorsqu'il en est trop éloigné, tout en lui permettant de garder le symbole dans le centre de son champ de vision.

Rhizome 2 : Scénario Déterministe - Naviguer de manière autonome à partir d'un plan de bâtiment

Comme précédemment, l'environnement d'exécution de Rhizome 2 est considéré déterministe. La connaissance *a priori* globale de l'environnement se présente sous la forme d'un plan papier présenté au robot au début de sa navigation. De manière similaire au processus cognitif d'un cerveau humain mis en œuvre lors d'un déplacement dans un bâtiment inconnu, Rhizome2 permet au robot (1) de «lire» le plan du bâtiment, (2) d'extraire et de «mémoriser» une séquence de symboles de navigation lui permettant d'atteindre sa destination finale, et enfin (3) de «reconnaître» en temps réel ces mêmes symboles dans l'environnement parcouru.

Un processus complet d'analyse du plan du bâtiment permet au robot d'extraire les informations pertinentes, soit ici, la séquence des symboles de navigation et leur signification correspondante (cf. module délibératif de la figure 4).

La structure neuronale construite autour de deux modules, respectivement, 1^{er} et 2nd module comportemental de la figure 2, permet la reconnaissance des symboles et l'apprentissage de l'association entre le symbole reconnu et sa signification, tout en donnant la possibilité au

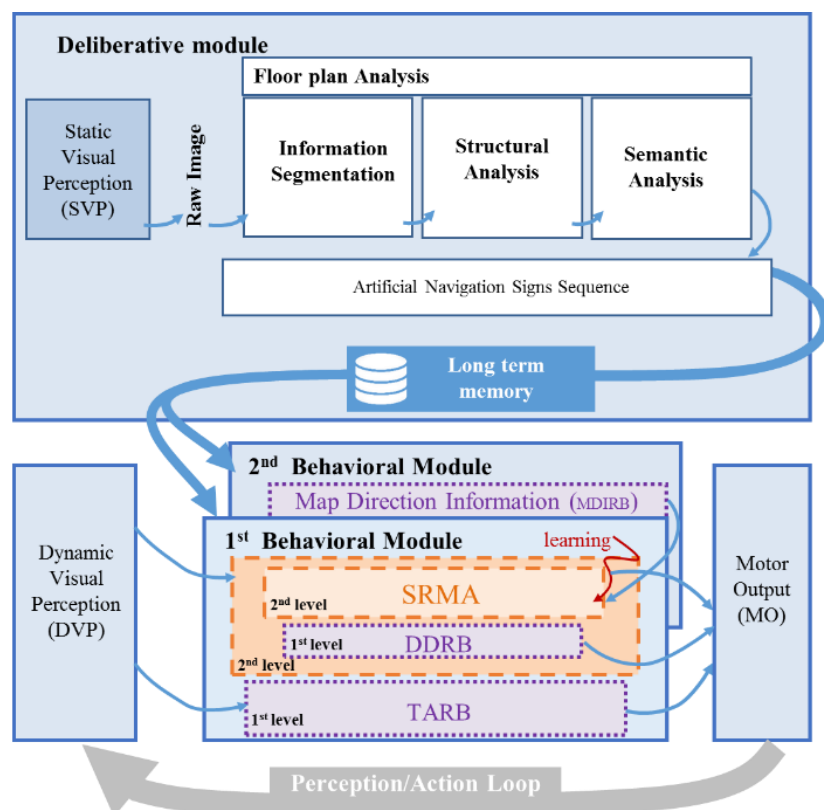


Figure 4. RHIZOME 2 composé d'un module délibératif en charge d'effectuer un processus d'analyse d'un plan de bâtiment et deux modules comportementaux. Le premier module comportemental correspond à celui présenté dans Rhizome1 et le deuxième permet d'obtenir la signification des symboles.

robot d'effectuer le mouvement associé directement.

Rhizome 3 : Scénario Stochastique - Apprendre et s'adapter en fonction des changements imprévus dans l'environnement

L'environnement d'exécution de Rhizome 3 est considéré comme dynamique, stochastique et donc plus réaliste. Rhizome 3 est conçu pour permettre au robot de faire face à tout changement non prévisible comme par exemple l'occlusion ou l'absence de symboles attendus (cf. Figure 5). Pour atteindre cet objectif, un système de reconnaissance de lieux construit sur un modèle de cellules de lieux est implémenté dans un troisième module comportemental. Un lieu est caractérisé par un ensemble de motifs, chacun associé à une position déterminée par rapport à un «nord» donné *a priori*. La robustesse de cette approche réside dans le fait que, même si un ou plusieurs motifs caractérisant l'endroit sont supprimés ou ne sont plus visiblement accessibles, un lieu peut encore être reconnu.

Par conséquent, alors que les deux premiers modules comportementaux sont en charge d'utiliser la séquence de symboles artificiels de navigation sauvegardée dans le module délibératif pour contrôler la navigation en ligne, un troisième module comportemental permet

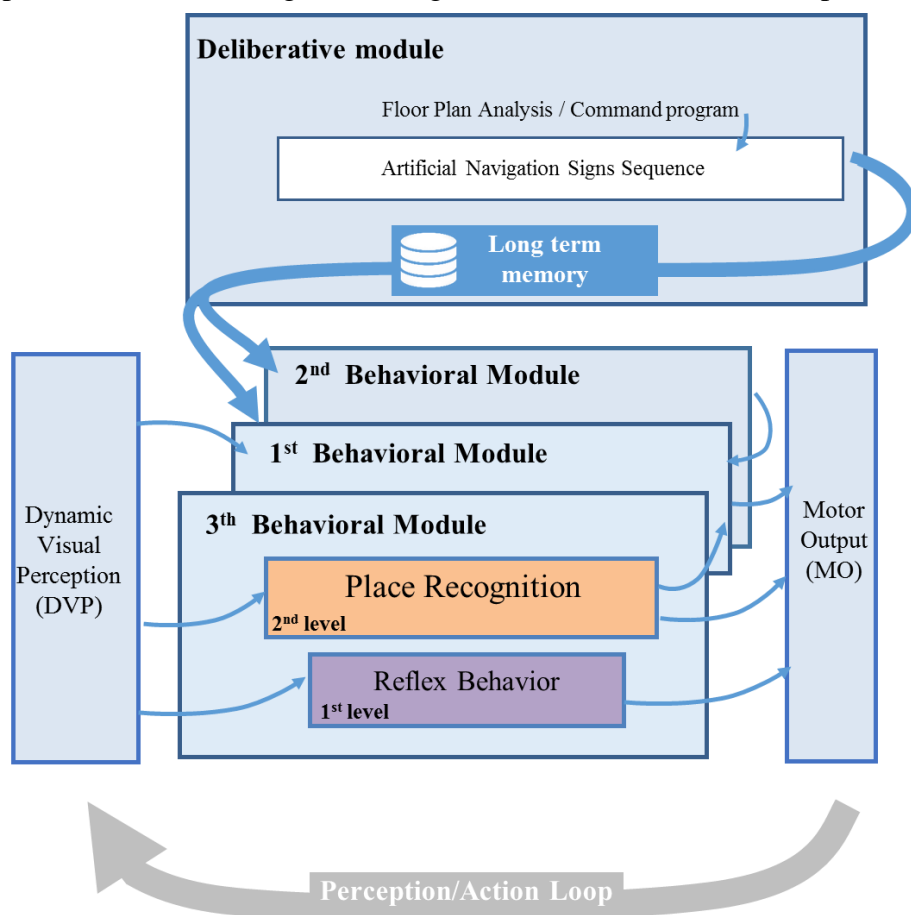


Figure 5. RHIZOME 3 est composé d'un module délibératif et de trois modules comportementaux. Le module délibératif sauvegarde et présente la séquence de symboles de navigation, apportée soit par un programme de commande, soit extraite d'un plan de bâtiment. Pour la navigation du robot, Les modules comportementaux utilisent, selon la situation rencontrée (présence ou absence de symboles artificiels), des informations visuelles, soit définies *a priori*, soit extraites dynamiquement. Dans ce dernier cas, un système de reconnaissance de lieux s'appuie sur cette information extraite pour définir de nouvelles positions de référence en remplacement des symboles.

au robot de trouver des lieux, de les apprendre et de les reconnaître comme de nouvelles positions de référence en remplacement des symboles.

L'ensemble du système fonctionne de manière parallèle et un «mécanisme compétitif» permet de décider du meilleur comportement (parmi les couches et les modules) pour le contrôle du robot en fonction du stimulus reçu. Ce sont les interconnexions neuronales composées de connexions excitatrices ou inhibitrices qui permettent ou annihilent l'activation des neurones. Lorsque l'apprentissage est nécessaire, un autre type de connexion est également utilisé: la connexion de modulation. L'apprentissage des associations entre le symbole reconnu et une action particulière est conditionné par un signal de renforcement qui représente les motivations internes du robot.

L'architecture RHIZOME : Une architecture de contrôle neurocomportementale hybride pour la navigation autonome indoor de robots mobiles reposant sur la perception visuelle

Résumé : Les travaux décrits dans cette thèse apportent une contribution au problème de la navigation autonome de robots mobiles dans un contexte de vision indoor. Il s'agit de chercher à concilier les avantages des différents paradigmes d'architecture de contrôle et des stratégies de navigation. Ainsi, nous proposons l'architecture RHIZOME (**R**obotic **H**ybrid **I**ndoor-**Z**one **O**perational **M**odule): une architecture unique de contrôle robotique mettant en synergie ces différentes approches en s'appuyant sur un système neuronale. Les interactions du robot avec son environnement ainsi que les multiples connexions neuronales permettent à l'ensemble du système de s'adapter aux conditions de navigation. L'architecture RHIZOME proposée combine les avantages des approches comportementales (e.g. rapidité de réaction face à des problèmes imprévus dans un contexte d'environnement dynamique), et ceux des approches délibératives qui tirent profit d'une connaissance *a priori* de l'environnement. Cependant, cette connaissance est uniquement exploitée pour corroborer les informations perçues visuellement avec celles embarquées. Elle est représentée par une séquence de symboles artificiels de navigation guidant le robot vers sa destination finale. Cette séquence est présentée au robot soit sous la forme d'une liste de paramètres, soit sous la forme d'un plan. Dans ce dernier cas, le robot doit extraire lui-même la séquence de symboles à suivre grâce à une chaîne de traitements d'images. Ainsi, afin de prendre la bonne décision lors de sa navigation, le robot traite l'ensemble de l'information perçue, la compare en temps réel avec l'information *a priori* apportée ou extraite, et réagit en conséquence. Lorsque certains symboles de navigation ne sont plus présents dans l'environnement de navigation, l'architecture RHIZOME construit de nouveaux lieux de référence à partir des panoramas extraits de ces lieux. Ainsi, le robot, lors de phases exploratoires, peut s'appuyer sur ces nouvelles informations pour atteindre sa destination finale, et surmonter des situations imprévues.

Nous avons mis en place notre architecture sur le robot humanoïde NAO. Les résultats expérimentaux obtenus lors d'une navigation indoor, dans des scénarios à la fois déterministes et stochastiques, montrent la faisabilité et la robustesse de cette approche unifiée.

Mots clés : *Architecture de contrôle neuronale robotique, navigation autonome indoor de robots mobiles, perception visuelle, fusion de données, analyse d'un plan du bâtiment, reconnaissance de symboles, approche hybride comportementale.*

The RHIZOME architecture: A hybrid neurobehavioral control architecture for autonomous vision-based indoor robot navigation

Abstract: The work described in this dissertation is a contribution to the problem of autonomous indoor vision-based mobile robot navigation, which is still a vast ongoing research topic. It addresses it by trying to conciliate all differences found among the state-of-the-art control architecture paradigms and navigation strategies. Hence, the author proposes the RHIZOME architecture (**R**obotic **H**ybrid **I**ndoor-**Z**one **O**perational **M**odule): a unique robotic control architecture capable of creating a synergy of different approaches by merging them into a neural system. The interactions of the robot with its environment and the multiple neural connections allow the whole system to adapt to navigation conditions. The RHIZOME architecture preserves all the advantages of behavior-based architectures such as rapid responses to unforeseen problems in dynamic environments while combining it with the *a priori* knowledge of the world used in deliberative architectures. However, this knowledge is used to only corroborate the dynamic visual perception information and embedded knowledge, instead of directly controlling the actions of the robot as most hybrid architectures do. The information is represented by a sequence of artificial navigation signs leading to the final destination that are expected to be found in the navigation path. Such sequence is provided to the robot either by means of a program command or by enabling it to extract itself the sequence from a floor plan. This latter implies the execution of a floor plan analysis process. Consequently, in order to take the right decision during navigation, the robot processes both set of information, compares them in real time and reacts accordingly. When navigation signs are not present in the navigation environment as expected, the RHIZOME architecture builds new reference places from landmark constellations, which are extracted from these places and learns the. Thus, during navigation, the robot can use this new information to achieve its final destination by overcoming unforeseen situations.

The overall architecture has been implemented on the NAO humanoid robot. Real-time experimental results during indoor navigation under both, deterministic and stochastic scenarios show the feasibility and robustness of the proposed unified approach.

Keywords: *Artificial neuronal network-based control architecture, autonomous mobile robot indoor navigation, visual perception, data merging, floor plan analysis, pattern recognition, hybrid behavior-based approach.*