Lambda-Field : a novel framework for risk assessment in occupancy grids

Johann Laconte

To cite this version:

Johann Laconte. Lambda-Field : a novel framework for risk assessment in occupancy grids. Automatic. Université Clermont Auvergne, 2021. English. NNT : 2021UCFAC085 . tel-03663791

HAL Id: tel-03663791
https://theses.hal.science/tel-03663791

Submitted on 10 May 2022

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 Clermont Auvergne
École Doctorale des Sciences pour l'Ingénieur

Lambda-Field: a Novel Framework For Risk Assessment In Occupancy Grids

---

Thèse présentée par **Johann Laconte**
Pour obtenir le grade de **Docteur d'Université**

Spécialité **Électronique et Systèmes**

En partenariat avec

Northern Robotics Laboratory, Université Laval, Canada

Institut National de Recherche pour l'Agriculture,
l'Alimentation et l'Environnement

Soutenue publiquement le **14 décembre 2021** devant le Jury composé de

Christian Laugier	Rapporteur	Directeur de recherche INRIA Grenoble
Ouiddad Labbani-Igbida	Rapporteur	Professeure des Universités Université de Limoges
Romuald Aufrère	Directeur de thèse	Maître de Conférences, HDR Université Clermont Auvergne
Philippe Bonnifait	Président du Jury	Professeur des Universités Université de Technologie de Compiègne
François Pomerleau	Examinateur	Professeur adjoint Université Laval, Canada
Roland Chapuis	Examinateur	Professeur des Universités Université Clermont Auvergne
Christophe Debain	Invité	Chargé de recherche, HDR INRAE Clermont-Ferrand

ACKNOWLEDGMENTS

Completing a Ph.D. is a long journey and one that cannot be attempted alone. During these years, one must seek guidance and comfort to overcome the numerous challenges a Ph.D. has to offer. As such, I had the opportunity to meet and discuss with a lot of incredible people who I would like to thank, as it is certain that the achievement of this doctorate is closely tied to them.

First, I would like to deeply thank my thesis director Romuald, whom I have known for more than eight years now. He has been a wonderful mentor during my whole university education, transmitting his passion for robotics, research and teaching. These three years working with him have been brilliant, as I enjoyed every discussion we had. To our future collaborations, as I am sure this is not the final chapter. In addition, I thank Roland for his advice and wisdom. I have sought to learn as much as possible from him during our many discussions. To Laurent, for his tremendous help with maintaining and operating the robots. I also had the opportunity to work with Abderrahim, whom I thank for the three years full of laughter spent together in our office. Many thanks to Elie and Ted who joined me in this journey, helping me in numerous aspects. This thesis would not be there without you.

This Ph.D. was a collaboration between several laboratories. As such, I had the opportunity to travel to Québec many times, each time being a precious opportunity to grow and learn from an amazing laboratory. I deeply thank François who accepted to collaborate with the Institut Pascal for this thesis, for his guidance and time, as the quality of the research done in this thesis was considerably increased thanks to him. Furthermore, I would like to thank the whole team, as I enjoyed every moment collaborating with them. More specifically, I thank Simon-Pierre, Dominic and Maxime for the precious time we spent together and the research that came out of these interactions.

I also collaborated with the National Research Institute for Agriculture, Food and Environment, where I had the opportunity to experiment our novel algorithms in unstructured environments. As such, I thank Christophe for taking the time to join, help and guide me in these experiments.

I would also like to thank the members of my jury for thoroughly reading the manuscript, for their invaluable feedback and the genuine interest they showed on my work. I would like to thank Philippe Bonnifait for cordially accepting to be the chairman of the thesis committee, as well as Christian Laugier and Ouiddad Labbani-Igbida for reviewing this manuscript.

This journey would not have seen any end without the support of my friends and family. A special thank to Mathieu and Simon for their help reviewing the articles I worked on, and for the thoughtful discussions we had that immensely helped me throughout this adventure. To our semi-philosophical debates, our far too long hikes that always lead to surprising lands, and the plentiful games of snooker we enjoyed on Sunday evenings. To all my colleagues for the laughter that helped reignite the fire during this long journey. To my family, whose unconditional support was the pedestal of this accomplishment. For that, many thanks to my mother, my father and my sister. Finally, I would like to thank my partner Mathilde for her precious encouragements. Her constant smile and positive spirit would brighten even the darkest of days.

ABSTRACT

In the context of autonomous robots, one of the most contentious topics is the notion of risk. Indeed, no robot escapes from such a question, that is whether robots will not cause any harm to themselves or the living beings in the surrounding environment. Robotics arms have a finite, relatively small workspace where the risk is tackled in a way that the robot completely stops whenever a human enters its workspace. In mobile robotics, this notion is more complex and still an open problem. First, a representation of the environment is needed for the robot to navigate in it. Oftentimes, the preferred representation of the environment is the semantic one, where each obstacle is stored as a single, unique entity. However, in complex scenarios or unstructured environments, detecting such obstacles is a tedious task and missing one could lead to disastrous events. In these cases, a metric map is used where each position stores the information of occupancy. The most common type of metric map is the Bayesian occupancy map. However, this type of map is not well-fitted to perform risk assessment for continuous paths due to its discrete nature. Hence, we introduce in this thesis a novel type of map called Lambda-Field, specially designed for risk assessment. The Lambda-Fields are a counterpart of the classical Bayesian occupancy grid. Instead of storing the probability of occupancy at each position, the Lambda-Field stores the intensity that a collision will occur at this position: the higher the intensity, the higher the probability of collision. Using this novel formulation, the Lambda-Fields are able to assess a generic risk over a path. Contrary to the Bayesian occupancy grid, the use of intensity instead of directly the probability of collision allows the risk assessment framework to produce physic-based metrics that conserve their physical units. Throughout this thesis, we present how to construct and use the Lambda-Field in both static and dynamic environments. We demonstrate that the Lambda-Field also possesses interesting mapping properties that induce more accurate maps of unstructured environments. Using this risk definition and the Lambda-Field, we show that our framework is capable of doing classical path planning but also cross unstructured environments where a Bayesian occupancy grid would not find any path.

Keywords: Mobile Robotics, Intelligent Vehicles, Risk Assessment, Occupancy Grids, Safe Navigation

RÉSUMÉ

Dans le contexte des robots autonomes, l'un des sujets les plus controversés est la notion de risque. En effet, aucun robot n'échappe à une telle question, à savoir si ce robot ne causera aucun dommage à lui-même ou aux êtres vivants dans l'environnement qui l'entoure. Les bras robotiques ont un espace de travail fini et relativement petit, où le risque est abordé de telle sorte que le robot s'arrête complètement dès qu'un humain pénètre dans son espace de travail. En robotique mobile, cette notion est plus complexe et reste un problème ouvert. Souvent, la représentation préférée de l'environnement est la représentation sémantique, où chaque obstacle est stocké comme une entité unique. Cependant, dans des scénarios complexes ou des environnements non structurés, la détection de tels obstacles est une tâche fastidieuse et en manquer un peut conduire à des événements désastreux. Dans ces cas, une carte métrique est utilisée, où chaque position stocke l'information d'occupation. Le type le plus courant de carte métrique est la carte d'occupation bayésienne. Cependant, ce type de carte n'est pas bien adapté pour calculer l'évaluation du risque pour les chemins continus en raison de sa nature discrète. Par conséquent, nous introduisons dans cette thèse un nouveau type de carte appelé Lambda-Field, spécialement conçu pour l'évaluation du risque. Les Lambda-Fields sont une contrepartie de la grille d'occupation bayésienne classique. Au lieu de stocker la probabilité d'occupation à chaque position, le Lambda-Field stocke l'intensité de la probabilité qu'une collision se produise à cette position : plus l'intensité est élevée, plus la probabilité de collision est élevée. Grâce à cette nouvelle formulation, les Lambda-Fields sont capables d'évaluer un risque générique sur un chemin. Contrairement à la grille d'occupation bayésienne, l'utilisation de l'intensité au lieu de la probabilité directe de collision permet à la méthode d'évaluation de risque de produire des métriques basées sur la physique qui conservent leurs unités physiques. Tout au long de cette thèse, nous présentons comment construire et utiliser le Lambda-Field dans des environnements statiques et dynamiques. Nous montrons que le Lambda-Field possède également des propriétés cartographiques intéressantes qui induisent des cartes d'environnements non structurés plus précises. En utilisant cette définition du risque et le Lambda-Field, nous montrons que notre méthode est capable de faire de la planification de chemin classique mais aussi de traverser des environnements non structurés où une grille d'occupation bayésienne ne trouverait pas de chemin.

Mots-clés : Robotique mobile, Véhicules Intelligents, Calcul de risque, Grilles d'Occupation, Navigation Sécurisée

CONTENTS

Acknowledgments **3**

Abstract **5**

Résumé **7**

Contents **9**

List of Figures **13**

List of Tables **17**

List of Acronyms **19**

1 General Introduction **21**
- 1.1 Overview 21
- 1.2 Proposed approach and contributions 23
- 1.3 Manuscript outline 25

I State of the art **29**

State of the art: Introduction **31**

2 How Should a Robot Assess Risk? **33**
- 2.1 Introduction 33
- 2.2 Definition of the Risk 34
- 2.3 What is a Good Risk Metric? 57
- 2.4 Conclusion 64

3 Mapping Frameworks **65**
- 3.1 Introduction 65
- 3.2 Semantic Approaches 66
- 3.3 Metric Approaches 73
- 3.4 Conclusion 90

4 Path Planning Algorithms 91

4.1 Introduction 91
4.2 Global Path Planning 93
4.3 Local Path Planning 99
4.4 Conclusion 105

Conclusion 107

II Proposed framework 109

Motivations 111

5 Static Lambda-Fields 113

5.1 Context of the work 114
5.2 Theoretical background 120
5.3 Proposed approach 121
5.4 Experimentations 134
5.5 Discussion 151
5.6 Conclusion 153

6 Dynamic Lambda-Fields 155

6.1 Context of the work 155
6.2 Proposed approach 158
6.3 Experimentations 167
6.4 Discussion 171
6.5 Conclusion 173

Discussion and Perspectives 175

III Appendices 179

Publications associated with this thesis 181

Other publications during the thesis 183

A Heterogeneous error regions 185

B Proof of equation 5.41 **187**

C Probabilistic error region **189**

D Computation of the confidence interval of the risk **191**

Bibliography **193**

LIST OF FIGURES

1.1 Example of Bayesian occupancy grid 23

2.1 Taxonomy of the risk 34

2.2 Example of unmanned robot in search and rescue environment 35

2.3 Example of DEM 36

2.4 Examples of sands on Mars 37

2.5 Example of hazardous terrain in the context of space robotics 37

2.6 Wavelet coefficients of a terrain 39

2.7 Tensor Voting illustration 41

2.8 Conditions for a robot to be in an admissible state 42

2.9 Lidar measurements during a heavy snowstorm 43

2.10 Terrain classification using a camera 45

2.11 Traversability estimation using super-pixels 45

2.12 Path planning on MRI 46

2.13 Example of hazardous urban scenario 47

2.14 Limitations of the TTC metric 49

2.15 Relation between the impact speed of the vehicle and the severity of the collision for the pedestrian 51

2.16 Example of terrain where deterministic prediction of the robot's displacement would fail 52

2.17 Convervative obstacle prediction approach 53

2.18 Example of learned traversability 54

2.19 Rover using efficiency-based risk metrics 56

2.20 Example of harmless robot 59

2.21 Example of scenario where the robot cannot guaranty its own safety . 61
2.22 Environment representation of a finite world 62
2.23 Illustration of the most used risk metrics 63
2.24 Illustration of navigation between randomly moving obstacles 63

3.1 Examples of semantic and metric maps 66
3.2 Example of urban environment 67
3.3 Example of lane detection 67
3.4 Example of OSM map 68
3.5 Example of clustering algorithm 69
3.6 Obstacle detection based on geometric models 70
3.7 Example of scenario where a clustering algorithm would fail to estimate the true shape of the obstacles 71
3.8 Example of incoherent results a neural network can yield 72
3.9 Taxonomy of the occupancy grid framework 74
3.10 Graph modeling of the occupancy map problem 75
3.11 Example of measurement model 77
3.12 Example of probabilistic quadtrees 79
3.13 Example of octrees 79
3.14 Illustration of the BOF 80
3.15 Example of BOFUM 81
3.16 Illustration of HS-BOF 83
3.17 Example of dynamic occupancy map from CMCDOT 83
3.18 Bayesian Network used by CMCDOT 84
3.19 Example of Gaussian process 85
3.20 Example of large environment mapped with Gaussian process 86
3.21 Amelioration of the Gaussian process framework 86
3.22 Comparison between maps produced with logistic regression and SVM 88
3.23 Illustration of learning approach in occupancy grids 88

4.1 Example of global planner and local planner 92
4.2 Example of Dijkstra algorithm 93
4.3 Example of A* algorithm 94
4.4 Example of D* algorithm 94
4.5 Graphical example of the ant colony algorithm 99
4.6 Example of state lattice with Reeds-Shepp trajectories 100
4.7 Example of node expansion 101
4.8 Example of local Rapidly-exploring Random Trees (RRT) 102
4.9 Example of clothoid tentacles 103

5.1 Example of unstructured environment where semantic maps would have trouble mapping all the potential obstacles 114
5.2 Example of occupancy grid the robot needs to cross 115
5.3 Example of collision assessment in an occupancy grid 116
5.4 Computation of the probability of collision in Lambda-Fields 122
5.5 Illustration of the error zone of the lidar sensor 122
5.6 Convergence of the confidence intervals 125
5.7 Example of lambda-Field the robot crosses 126
5.8 Decomposition of the robot’s speed 128
5.9 The robot crosses an area with tall grass 129
5.10 Examples of probability density function of several labels 130
5.11 Robot used in the experimentations 135
5.12 Convergence of the collision probability for a given cell occupied at 40 % and 60 % 138
5.13 Evolution of the collision probability of a fully occupied cell that has been wrongly measured empty 139
5.14 Mapping environment, consisting of a spare wire fence with some tall grass on the left, as well as a small truck 140
5.15 Mapping of a wire fence 141

5.16 Mapping error of the wire fence for the Bayesian occupancy grid and the Lambda-Field 141

5.17 Aerial View of the mapped environment 142

5.18 Bayesian occupancy grid and Lambda-Field of the urban environment 143

5.19 Difference of lambdas between the expected lambdas and the confidence interval for a structured environment 143

5.20 Unstructured environment used for the mapping experimentation . . 144

5.21 Mapping of an unstructured zone 145

5.22 The robot has to avoid a tree 145

5.23 Lambda-Field of the tree environment 146

5.24 Speed and risk of the chosen paths going around a tree 147

5.25 The robot has to reach a goal behind the tall grass 148

5.26 Lambda-Fields of the tall grass environment 149

5.27 Speed and risk of the chosen path going around the tall grass 149

5.28 Speed and risk of the chosen path going through the tall grass 150

6.1 Example of an urban scenario that the robot can encounter 156

6.2 Example of collision probability prediction for a vehicle 157

6.3 Example of Dynamic Lambda-Field where a pedestrian emerges unexpectedly on the road 160

6.4 Example of velocity estimation 165

6.5 Example of risk assessment 167

6.6 Convergence of the speed and orientation of a single dynamic obstacle 168

6.7 Convergence of the class probability for a pedestrian and a car 169

6.8 Example of field resulting from the Dynamic Lambda-Field 171

6.9 Risk undergone by the robot during its traversal with the associated speeds 172

LIST OF TABLES

1.1 Outline of the main differences between the standard Bayesian occupancy grid and the Lambda-Field. 25

2.1 Excerpt of works tackling traversability analysis with geometric methods 44

2.2 Excerpt of works tackling traversability analysis with appearance-based methods 46

2.3 Some examples of TTC measures 50

3.1 Comparison of the metric mapping methods 89

4.1 Main sample based planning algorithms 97

4.2 Comparison of the main planning techniques 106

LIST OF ACRONYMS

BEV	Bird Eye View	73
BOF	Bayesian Occupancy Filter	31
BOFUM	Bayesian Occupancy Filter Using prior Map knowledge	81
BOG	Bayesian Occupancy Grid	75
CNN	Convolutional Neural Network	69
CMCDOT	Conditional Monte Carlo Dense Occupancy Tracker	83
CRF	Conditional Random Field	69
CVaR	Conditional Value at Risk	63
DARPA	Defense Advanced Research Projects Agency	67
DEM	Digital Elevation Map	36
DST	Deceleration to Safety Time	51
EST	Expansive Space Trees	97
lidar	Light Detection And Ranging	36
GESTALT	Grid-based Estimation of Surface Traversability Applied to Local Terrain	41
GP	Gaussian Process	84
GPU	Graphical Processing Unit	71
HD	High-Definition	69
HM	Hilbert Maps	89
HMM	Hidden Markov Model	54
HS-BOF	Hybrid Sampling Bayesian Occupancy Filter	82
ICP	Iterative Closest Point	75
MPC	Model Predictive Control	52
MRI	Magnetic Resonance Image	46
NODR	Negative Obstacle DetectoR	41
OCR	Optical Character Recognition	72
OSM	OpenStreetMap	67
PCIR	Probability of Collision with Injury Risk	53

PET	Post-Encroachment Time	49
PRM	Probabilistic Roadmap Method	96
radar	Radio Detection And Ranging	82
RPP	Randomized Potential Planner	97
RRT	Rapidly-exploring Random Trees	15
RSS	Responsibility-Sensitive Safety	61
SMC-BOF	Sequential Monte Carlo Bayesian Occupancy Filter	82
SVM	Support Vector Machine	41
TTC	Time To Collision	49
TTR	Time To React	51
UAV	Unmanned Aerial Vehicle	75
YOLO	You Only Look Once	72

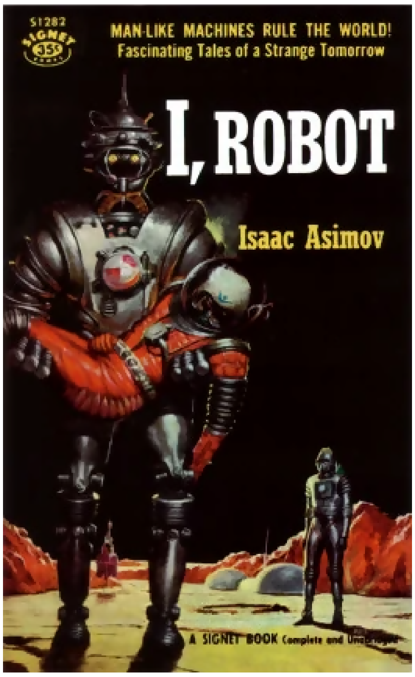
CHAPTER 1

GENERAL INTRODUCTION

1.1 Overview 21
1.2 Proposed approach and contributions 23
1.3 Manuscript outline 25

1.1 Overview

Over the past decades, robotics systems started to prove themselves useful in numerous applications, ranging from environment monitoring, mines inspections, search and rescue missions to intelligent transportation systems. In the one hand, robots are a fantastic replacement for dangerous, tedious jobs such as mining operations. On the other hand, they can also help and replace activities prone to fatal mistakes such as driving, yielding in the long term to safer transportation systems. As the robots become more and more a part of our everyday life, accidents and risky events and more and more prone to occur. These situations can happen for numerous reasons, extending from hardware failures such as processors errors due to radiations to collisions with people due to excessive noise in the measurement data.


S1282
SIGNET
35¢
MAN-LIKE MACHINES RULE THE WORLD!
Fascinating Tales of a Strange Tomorrow
I, ROBOT
Isaac Asimov
A SIGNET BOOK Complete and Unabridged

Although we believe there is still a long road ahead for the robots to work in the overall complexity of the world, the imaginary of people conceptualize robots and their impact of society long before the robots actually started to prove themselves useful. Russian-born American science-fiction writer Isaac Asimov first used the word ‘Robots’ in 1942 in his short story ‘Runabout’. The name ‘Robot’ itself comes from a 1920 Czech-language play, denoting at the time a fictional humanoid. In this book, he proposed that the robots have to follow, in this order, the three following rules:

First Law A robot may not injure a human being or, through inaction, allow a human being to come to harm.

Second Law A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

Third Law A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Of course, such laws are easier to write than actually coding them on a computer. However, one key aspect of the robotics is retrieved in these laws, that is the guaranty that the robot will not harm others or himself. With reasons, we can easily notice from the ongoing pop-culture that the humanity is somewhat conflicted between two visions of the robotics. Although robotics promises to revolutionize numerous domains and save uncountable lives in the future, there is the risk that these robots also unintentionally harm others. As such, laws have been erected to prevent this, or, in the current state of the robotics, framework have been built to provide guaranties of safety.

The context of this Ph.D. thesis established itself deep within these considerations. Indeed, plentiful questions remain at least partially unanswered in the literature: What is ‘safe’? In the scenario where the robot has to choose between two unsafe actions, which one should it chooses? On which criteria? Will the robots ever able to guaranty their own safety or does this notion necessarily rely on others’ choices?

In this thesis, we worked at proposing a novel framework for risk assessment for mobile robotics that is heavily generalizable. Indeed, as we show in the next chapters, the notion of risk or safety is not unique and a unique definition cannot suit every applications robotics has to offer. Using our framework, the robot is able to infer the risk of a given path and therefore makes more informed decisions, knowing the risk tied to every of its possible actions.

1.2 Proposed approach and contributions

More precisely, we propose a novel mapping framework, called Lambda-Field, that provides natural ways to infer risks for a given path in the environment. The risk stays generic in the entire mathematical derivation and is only implemented during the experimentations phase, meaning that the risk can be adjusted depending on the application without the need to change the mathematical theory behind it.

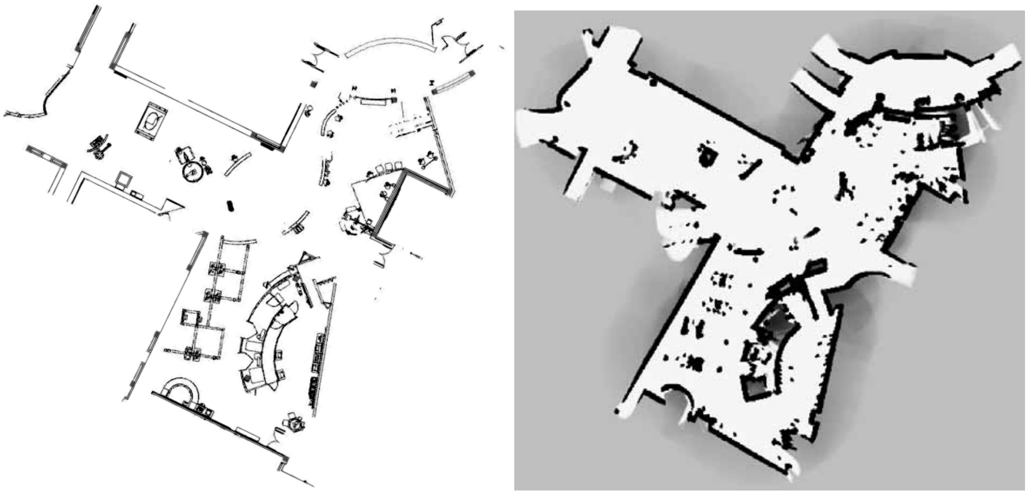


Figure 1.1: Example of Bayesian occupancy grid, from [1]. Left: Architectural blueprint of a large open exhibit space. Right: Bayesian occupancy grid of the same space. The darker the cell, the higher the probability of occupancy.

In order to provide a conservative approach, we chose to use occupancy grids to represent the environment, where the Bayesian occupancy grid is the most popular implementation. Figure 1.1 provides an example of such an occupancy grid for an office space. The Bayesian occupancy grid saves a tessellated environment where each cell stores the probability of occupancy of the underlying space, where 0 denotes an empty space (i.e., a probability of occupancy of 0) and 1 a certain obstacle at this position. However, as studied in this manuscript, this framework is not able to infer meaningful, continuous risks as the framework was simply not designed for such a task. As such, risk assessment on Bayesian occupancy grids can yield counter-intuitive behaviors depending of the way authors infer risks on it, potentially leading the robot into risky paths for either itself of the other agents of the environment. Therefore, we propose a counterpart of this occupancy theory to infer meaningful risks in occupancy grids that does not suffer such drawbacks. Our main contributions are

A novel mapping framework, called Lambda-Field, that is specially conceived to infer risks in occupancy grids. In contrast to the Bayesian occupancy grid, the Lambda-Fields do not store the probability of collision at each position but an intensity that can be seen as the 'density' of the event 'collision' at this position. The higher the intensity, the more likely a collision will happen at this position. Using this framework, one can infer the probability of collision for a path by integrating the intensity over the path. Moreover, more complex risks can be inferred without any change in the theory. The risk function can better characterize the hazardousness of a path, as for instance colliding with a car at low speed is indubitably safer than colliding with a pedestrian at high speed.

A comparison with the Bayesian occupancy grid: We provide a theoretical and experimental comparison with the classical Bayesian occupancy grid for static environments. More specifically, we show that the Lambda-Fields are better suited to store unstructured obstacles (e.g., bushes, wire fences), where the Bayesian occupancy grid would wrongly converge in specific cases. Furthermore, we investigate the recovery rate of the frameworks, namely the speed at which one framework can recover from a wrong estimation of a cell value and re-converge to its true state, showing that each framework has its pros and cons.

A risk assessment method: Finally, we use our novel mapping framework to perform path planning in both structured and unstructured environment, for static and dynamic cases. These experimentations in real-world conditions allow us to effectively see the practical differences the Lambda-Fields bring. In the static case, the Lambda-Fields allow the robot to cross unstructured obstacles such as tall grass while being aware of the potential risk of such actions, whereas in the dynamic cases more informed decisions can be made.

Table 1.1 provides a summary of the main differences between the Bayesian occupancy grid and the Lambda-Field investigated in this manuscript. Whereas the Bayesian occupancy grid is based on Bayesian inferences, the Lambda-Field bases itself on the Poisson Point Process. Using this process, the stored information is no longer a probability of occupancy but the intensity of a given event that we define here as a collision with an obstacle. This formulation allows the computation of probabilities of collision for a given path, namely a subset of the environment, in contrast to the Bayesian occupancy grid that gives the probability of collision for a given position. Finally, we will demonstrate later on that for a given cell, the Lambda-Fields can converge in the whole range of collision probability $[0;1]$ whereas the Bayesian occupancy grid always converges to either 0 or 1. Furthermore, the Lambda-Field recovers in a polynomial manner whereas the Bayesian occupancy grid has a logistic shape (i.e., slower than polynomial). In a more practical point of view, this means that the Lambda-Field is quicker to take into account the new information and recover but is slower on the long term to completely erase the false information.

	Based on	**Stores**	**Infers**	**Occupancy convergence**	**Error recovery**
Lambda-Field	Poisson Point Process	Intensity of the event 'collision' $\lambda \in [0;\infty)$	Probability of collision for a given subset $\mathscr{P} \subset \mathbb{R}^n$	$[0;1]$	Polynomial
Bayesian Occupancy Grid	Bayesian Inference	Probability of collision $p \in [0;1]$	Probability of collision for a given position $\mathbf{x} \in \mathbb{R}^n$	0 or 1	Logistic

Table 1.1: Outline of the main differences between the standard Bayesian occupancy grid and the Lambda-Field.

1.3 Manuscript outline

In this manuscript, we present our framework called Lambda-Field. In a global manner, we first try to give an overview of risk management in robotics, that ranges from answering the question ”what is a risk?” to ”How to plan risk-aware trajectories?”. Then, we present our developed framework, starting with the theory in the static cased, extensively comparing it to the state of the art and showing its advantages. Then, we propose to extend it to the dynamic realm and once again show its utility for risk management in accident mitigation scenarios. More specifically, the manuscript is arranged as follows:

Part I provides a state of the art of the robotics work revolving around risk assessment.

Chapter 2 gives an overview of how the risk is perceived in robotics for diverse applications. Indeed, the notion behind this generic word is very different depending on the context, that is either space robotics or intelligent transportation systems, for instances. After a presentation of the main used metrics, we present some frameworks that aim at generalizing the notion of risk for a broader class of robotics systems. This problem raise several interesting questions from both a philosophical and practical point of view: 'are every risk metrics equal?', 'How can we translate an ethical theory to a practical metric?', 'Can a robot always guaranty its own safety?' and so on.

Chapter 3 provides a survey of the different mapping techniques, with a focus on the metric approaches. Indeed, one cannot infer risk without at least a way to store its surrounding environment. First, we present the semantic approaches that are the preferred ones for risk assessment. However, as argued in this chapter, such maps are not easy to build and in a conservative manner, metric maps are far easier to provide. Thus, we focus this chapter on metric maps and present in depth the theory behind the Bayesian occupancy grid that is the most popular metric map framework. We also present an extensive survey of the extensions of the former theory where numerous works have been built upon the framework.

Chapter 4 presents the main planning frameworks for robotics systems, with a highlight on the ones used in intelligent transportation systems. We start by presenting the global path planning algorithms which aim at finding a trajectory to a far away goal without considering much the kinematic constraints of the vehicle nor the local risk. Then, we look at the local planners which focus on following the global trajectory while taking into account local constraints such as the risk or the ongoing dynamics of the robot.

Part II presents our framework. With the knowledge of how to assess risk, how to map and how to safely plan trajectories, we argue the need of a framework able to infer meaningful risks in metric maps and provides them to a path planning algorithm.

Chapter 5 provides the core theory of the framework, starting from the mathematical derivations to real-world experimentations showing the applicability of the theory. The framework is compared to the Bayesian occupancy grid, showing the main differences and its advantages in the context of unstructured environment. Using our framework, safer maps of unstructured environments are built and thus safer and more informed decisions can be made by the robot.

Chapter 6 extends the former framework to the dynamic realm in the context of urban navigation. Indeed, as argued throughout this manuscript, risk assessment on Bayesian occupancy grids can yield incoherent decisions from the robot. Henceforth, we show that using our framework and an implementation of the risk function as the maximum gain of kinetic energy resulting from a collision, the robot makes coherent decisions and can prefer a more dangerous collision if that saves a pedestrian, for instance.

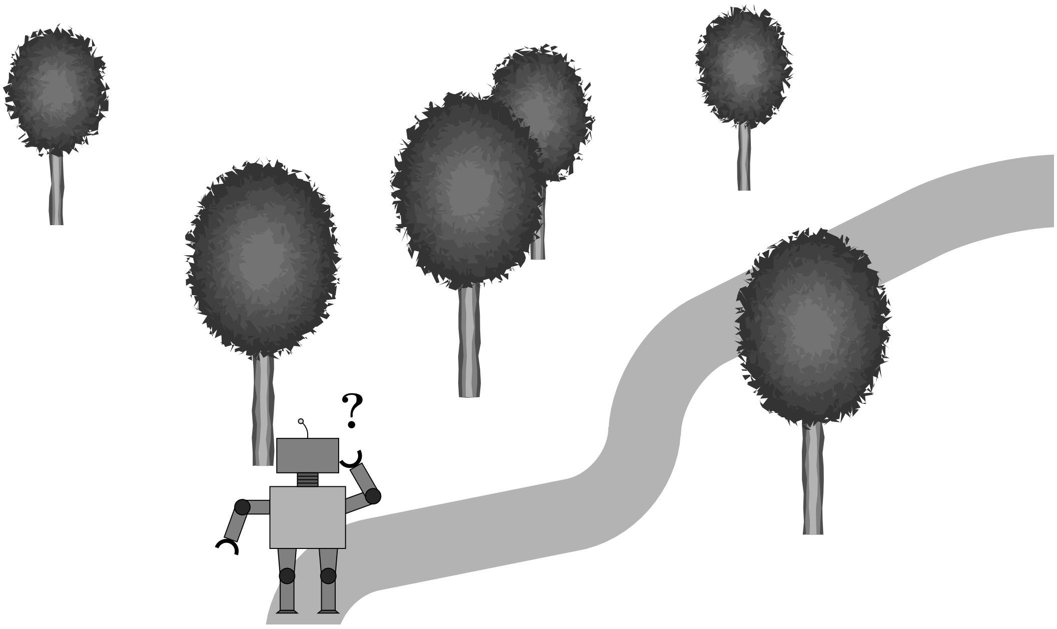
Finally, we provide a general conclusion of our framework, discussing that it is heavily generalizable and that several works already aim at generalizing both the mapping and planning process linked to the risk management.

Part I

State of the art

STATE OF THE ART: INTRODUCTION

In this part, we present an overview of the tools a robot needs to evolve *safely* in a complex environment. First, we look at how the word 'safety', in other word the *risk*, is defined in robotics. Depending on the application, this word can hide very different meanings that are not compatible every time. In addition to providing a taxonomy of the different risk metrics, we propose a small discussion about what the risk should represent. This leads to taking a higher vision of the concept, where some authors tried to unify the different frameworks and propose a generalized notion of risk. Then, we present a comprehensive study of the mapping techniques. Indeed, no robot can evolve nor infer risks without an internal representation of the environment. Particularly, we focus on the metric maps and the very popular Bayesian Occupancy Filter (BOF). Finally, we give an introduction to the different planning techniques. Without details, every main technique is presented with a few examples of authors that took into account the risk into their planning. Using these three parts, a robot should be able to effectively perceive and plan trajectories while being aware of the underlying risk taken at each decision.



CHAPTER 2

HOW SHOULD A ROBOT ASSESS RISK?

2.1 Introduction **33**
2.2 Definition of the Risk **34**
  2.2.1 Risk As a Traversability Analysis 34
  2.2.2 Risk As a Collision Analysis 47
  2.2.3 Risk As an Efficiency Analysis 55
2.3 What is a Good Risk Metric? **57**
  2.3.1 The contradiction of the utilitarianism 57
  2.3.2 Unification frameworks 58
2.4 Conclusion **64**

2.1 Introduction

In the robotics literature, the notion of risk finds its place in almost all possible applications. Behind this generic word lies a concept deeply embedded in robotics: what is safe for the robot?

In this chapter, we will look at how the state of the art defines the risk and thereby see that there is not one but many definitions. Indeed, the risk defined for a rover in the surface of Mars is not the same as for an autonomous car. In the first case, the rover genuinely needs to monitor its power consumption and whether the environment in front of it is traversable. The risk here is that the robot gets stuck in the mud or simply runs out of power in a hazardous environment such as a sandstorm. However, the risk for an intelligent vehicle is quite different. Even though it still has to watch out for traversable path, this is often not an issue as the roads are made to be traversable by such machines. In the context of autonomous vehicles, the main hazard is a collision with the environment or other agents such as a car or a pedestrian. Therefore, the risk defined in these applications is directly linked to the probability of collision.

Using these few examples, we propose to divide the large notion of risk into three main components, as shown in Figure 2.1. At first, we analyze the risk based on traversability, used in space and agricultural robotics for instance. Then, we look at the risk coming from collisions where the rise of intelligent vehicles leads to a great number of interesting theories. Finally, we investigate a few examples of risk defined as the efficiency of the task.

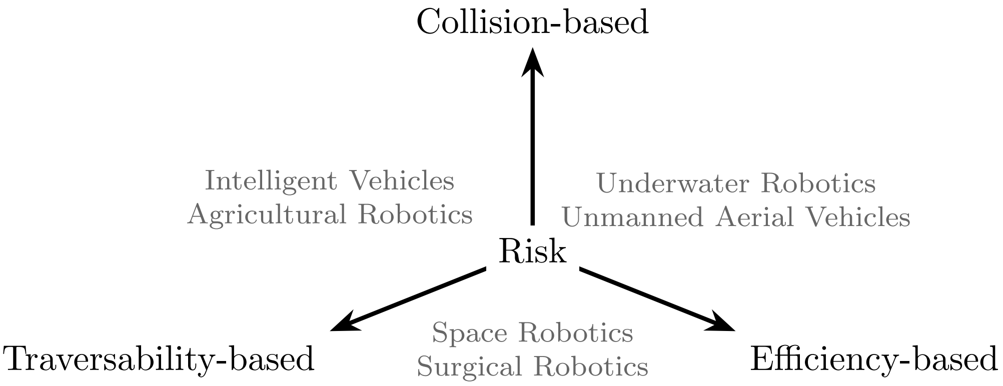

Collision-based
Intelligent Vehicles
Agricultural Robotics
Underwater Robotics
Unmanned Aerial Vehicles
Risk
Traversability-based
Space Robotics
Surgical Robotics
Efficiency-based

Figure 2.1: Taxonomy of the risk with examples of robotic applications, divided into three main branches that are the traversability, collision and efficiency.

Once the risk has been presented in various domains, we will discuss more in depth of what the risk is and how to quantify it. Indeed, in this vast ocean of definitions, one can easily get lost. Some authors proposed some theories to unify the definitions, better define what is a risk and how to assess it. This leads to a discussion of what a risk metric should represent, philosophically and mathematically speaking.

2.2 Definition of the Risk

2.2.1 Risk As a Traversability Analysis

Motion planning constitutes a domain where several disciplines meet such as artificial intelligence, robot perception and computer vision. In light of the plurality of the applications of mobile robotics, the notion of whether the robot can safely navigate in an environment does not have a single definition. In this section, we present the notion of traversability for mobile robots in static environments. Traversability analysis finds its root in numerous domains such as search and rescue mission, agricultural robotics or space robotics.

In contrast to more well-defined robotics applications where robots are designed to evolve indoor, the notion of traversability finds its meaning where the terrains the robots evolve in might be unsafe. For instance, in the case of search and rescue missions, the environment contains a lot of wreckage, negative obstacles and possible hazardous threats like fire or loose electrical wires that can directly jeopardize the integrity of the robot. Figure 2.2 gives an example of environment a search and rescue robot might need to cross. In space robotics, the main hazard is the robot getting stuck in loose soil, leading the mission to fail since it is for now ill-advised to go to mars and give it a hand.

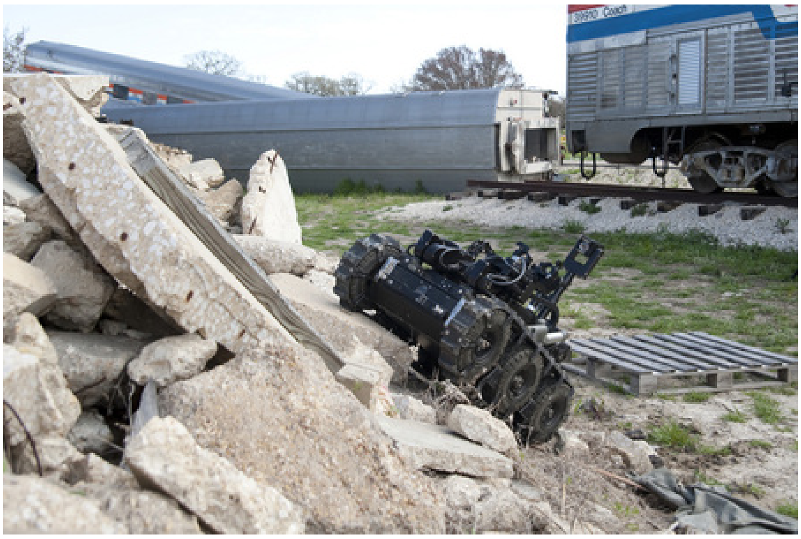


Figure 2.2: Example of unmanned robot in search and rescue environment containing many hazards such as holes and loose wires, from [2].

In the following, we define the risk as a traversability metric, as done in [3]:

Traversability The capability of a ground vehicle to reside over a terrain region under an admissible state wherein it is capable of entering given its current state, this capability being quantified by taking into account a terrain model, the robotic vehicle model, the kinematic constraints of the vehicle and a set of criteria based on which the optimality of an admissible state can be assessed.

Thus, this formulation only takes into account the interaction between the robot and the environment. However, in the context of static environment, no other agent can harm the robot, therefore this definition suffices for the time being.

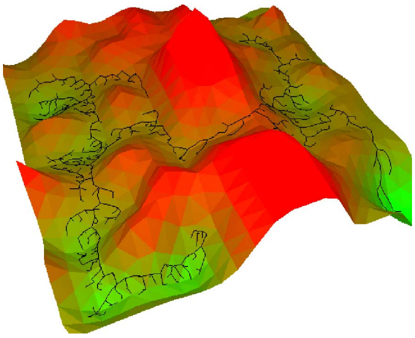


Figure 2.3: Example of Digital Elevation Map (DEM) where the color corresponds to the elevation, from [4].

The notion of traversability arose first with the conception of 2D Digital Elevation Map (DEM) [5], alternatively known as Cartesian elevation maps [6], [7]. This was the direct extension of the occupancy grids introduced by Elfes [8], where a more thorough description of his work is given in Chapter 3. The advantages of grid-based terrains, where an example is depicted in Figure 2.3, are that they enable rapid and efficient graph search algorithms. Such methods are still widely used nowadays, such as [4], [9]–[13].

The construction of DEM is preferred when range sensors are used such as Light Detection And Ranging (lidar). However, other sensors' modalities might also provide useful information, such as a camera or directly the proprioceptive information of the robot.

Henceforth, we split this section into three main parts. First, we investigate the traversability analysis using proprioceptive analysis. Namely, what are the information that the robot can recover using proprioceptive sensors such as IMU, odometry and so on. The second and third parts deal with exteroceptive analysis, meaning that the robot aims to estimate the traversability of the terrains without actually crossing it. We investigate at first traversability analysis using a range sensor, leading to a geometry-based approach. Finally, we look at the traversability analysis with appearance-based methods, using spectroscopic sensors such as a camera.

Proprioceptive Traversability Analysis

Proprioceptive analysis of the traversability is very important to learn the slipping and skidding models and therefore infer whether a terrain is traversable. Indeed, such information is essential for accurate motion planning [14]. As mentioned by Baril *et al.* [15], the accuracy of the modeling of the robot's dynamics highly depends on the soil parameters and can be learned while the robot crosses the environment. Furthermore, the traversability information for each type of terrain can be propagated for unknown regions using long range sensors. As an example, Cunningham *et al.* [16] predicted the slip parameter using Gaussian Processes in the context of space robotics, using proprioceptive information as well as a camera. Figure 2.4 shows the Marsian rover Curiosity with different types of soils that may be hazardous. As such, it is essential to estimate and predict which soil will be the safest to cross.

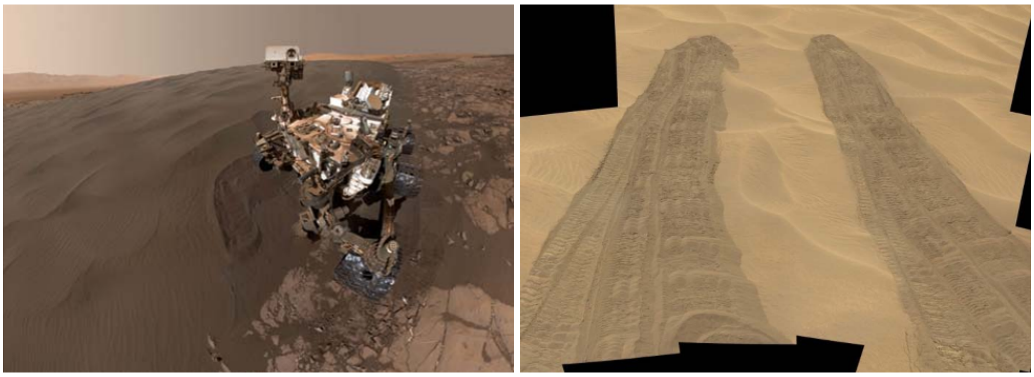


Figure 2.4: Examples of sands on Mars with very different visual appearances. Left: Curiosity in the Bagnold Dunes. Right: Wheel slip of Curiosity in the Hidden Valley. Image credit: NASA/JPL.

In the context of space robotics, the main hazard is the robot to get stuck in loose soil. As shown in Figure 2.5, a rover can easily become trapped in loose soil, leading to the failure of the mission. In this example, the rover Opportunity dug more than 10 cm deep into the soft and sandy material of Mars' Meridiani Planum region during the rover's 446th martian day. More than five weeks were needed for the team to test, plan and monitor the escape of this deadly sand. In that sense, many works aimed to better understand and predict robot-soil interactions. Shneier *et al.* [17] proposed a method to learn and associate traversability to regions of the environment, using a fusion of external and internal sensors. Brooks *et al.* [18] analyzed the ongoing vibrations induced in the robot structure by the wheel-terrain interaction to classify the type of soil the rover is currently crossing. A vibration sensor was mounted on the wheels, allowing the robot to classify between gravel, Mars-1 Soil Simulant [19] and washed beach sand. Garcia Bermudez *et al.* [20] investigated the performance of a legged robot in distinctive rough terrains (tile, carpet and gravel), and classified the terrains using vibration data. A comparison of classifier performance for vibration based terrain classification can be found in [21]. In the same fashion, Ojeda *et al.* [22] classified several terrains using a neu-

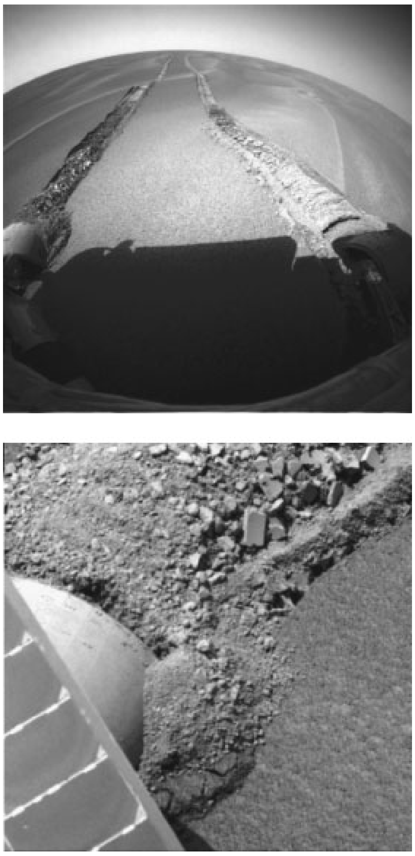


Figure 2.5: Example of hazardous terrain in the context of space robotics. The rover Opportunity is trapped in the Purgatory dune on sol 447. More than five weeks were required to get the rover out of the dune. Image credit: NASA/JPL.

ral network that feeds from different sensors sources. Angelova *et al.* [23] measured the slip the robot was currently facing to learn a correlation between the terrain appearance, geometric information and the predicted slip. This method is intended to improve navigation for rover in steep slopes and rough terrains for Mars rovers.

In another field, the notion of traversability is also deeply connected to agricultural robotics. Because of the often massive vegetation hiding potential obstacles, the evaluation of the safety of potential actions is challenging. Wellington *et al.* [24] used a learning approach to teach a tractor to differentiate between safe vegetation and hazardous obstacles such as wall or humans. Leppänen *et al.* [25] developed a sensing method to determine the quality of a terrain while driving. Using a mobile robot that can measure the vertical, horizontal and rotational strain forces affected its wheels, they were able to compute various parameters linking the interaction between the robot and the terrain. Such parameters can then be used to train a classifier and associated them with image or range data. Bajracharya *et al.* [26] used a variety of sensors to adapt to local terrains while extend proprioceptive information into the far field, avoiding the 'myopic' behavior. Using sparse proprioceptive examples, the robot was able to teach a short range geometry-based terrain classifier as well as a long range image-based classifier, itself learning from the short range one. In the same fashion, Howard *et al.* [27] proposed a method for a robot to learn traversability from experience, and more precisely from 3D geometry and proprioceptive information. Field tests show that the robot was effectively able to learn the traversability of the vegetation terrains and to extend its knowledge using a vision system.

Although the aforementioned methods effectively estimate the traversability of the terrains, it requires the robot to actually take risks by engaging this route. Hence, this requires to have highly tolerant, maybe disposable robots to deal with the coming failures and crashes. As it is not often a possibility to lose robots, the next sections present methods to predict as precisely as possible the traversability of the surrounding environment without having to go through it.

Geometry-based Traversability Analysis

The vast majority of traversability estimation methods relies on geometric processing. In the set of large trends that have been explored, one can cite the main approaches that are

- Signal processing, assuming that the terrain is a smooth 2D signal; and
- Statistical processing and extraction of moments-based features.

Other the next paragraphs, we will briefly analyze each main trend in traversability analysis.

Signal processing methods Although the signal processing methods are not the most popular, they still constitute an interesting field to explore. One of the earliest works on signal processing for traversability assessment is the work from Hoffman *et al.* [28], where the roughness of the terrain is extracted using Fourier analysis. Lu *et al.* [29], [30] also used Fourier analysis with a laser stripe-based structured light sensor, classifying the terrain in front of the robot. Pai *et al.* [31] used wavelet decomposition to model the terrain in different resolution levels. In rough areas, high frequency wavelets are needed to represent the terrain. Using this observation, they define a roughness measure based on the vanishing of the wavelet coefficients (from coarse to fine). Figure 2.6 shows an example of wavelets analysis. For smoother terrain, the wavelet coefficients quickly decrease to zero, whereas tougher terrains need finer wavelets to be represented.

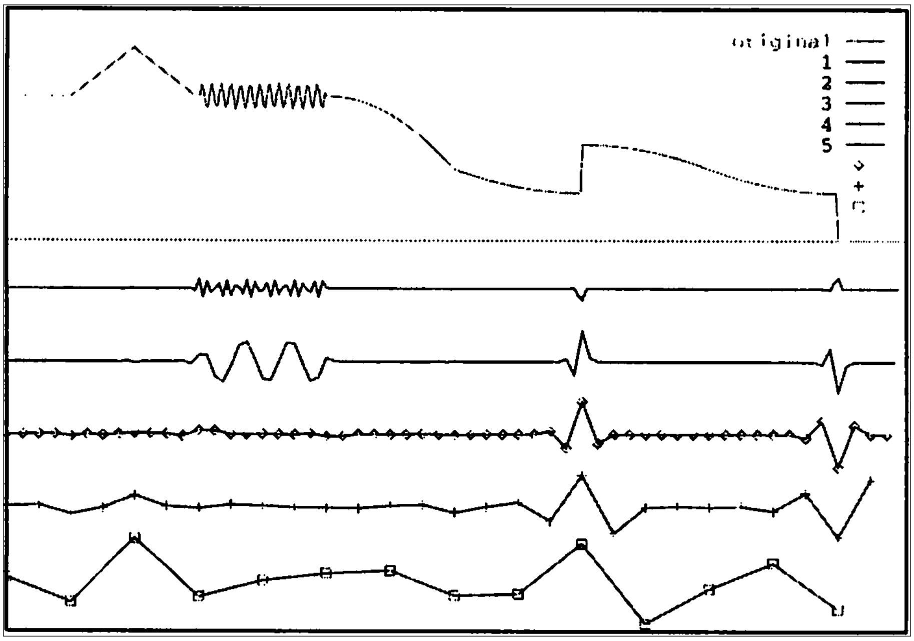


Figure 2.6: Terrain (Top) and its wavelet coefficient levels (Bottom), arrange from fine (higher) to coarse (lower), from [31]. Tougher terrain leads to higher coefficients for finer wavelets, whereas these coefficients quickly vanish for easily traversable terrains.

Statistical processing Statistic processing of the underlying 3D information of a terrain has been the most popular and most explored area in the domain of the traversability. The core concepts go back to the works of Langer *et al.* [32] and Gennery [33]. Langer *et al.* [32] constructed traversability maps by computing elevations statistics of the terrain for each cell of the map, namely the minimum and maximum height, the variance and the slope. The features were then compared to hard thresholds according to the robot's capabilities (e.g., the maximum height it was able to climb safely). On the contrary, Gennery [33] proposed to aggregate the different features into a unique cost function that was then used by a path planner algorithm. Building on their works, numerous works have emerged during the following years. In [34], a traversability index was defined for each cell, taking into account the roughness and slope of the terrain. This index was used to construct a so-called ‘polar traversability index‘ that represents the overall difficulty of traveling along the corresponding direction. In another fashion, Andersen *et al.* [35] chose to use a 2D lidar inclined toward the ground to extract local features (height, roughness, step size, curvature, slope, width and data validity) and fed them to a classifier. Along with these techniques, most of the works use of a convolution kernel to simulate the effect of the terrain on the robot. The map is convolved with the footprint of the robot, hence simulating the overlay of the vehicle on top of the terrain. Features representing the traversability at this configuration are then computed for each robot’s position.

Roboticists discovered quite quickly that deterministic systems were not able to tackle every problem robotics has to offer. In that sense, a number of approaches focused on modeling uncertainties in terrain perception and traversability estimation. The foundations in this field were brought by Kavraki *et al.* [9] and Gennery [33]. Singh *et al.* [36] modeled errors and uncertainties in their work, introducing the notions of ‘certainty’ and ‘goodness’ to create traversability maps. The goodness of a cell was determined by the suitability for travel, whereas the certainty was simply the confidence over the shape of the terrain in the cell.

Another interesting approach, instead of extracting basic statistics for each cell, is to look at the shape of the terrain. As highlighted by Labussière *et al.* [37], a point cloud carries a lot of information and shapes such as edges, surfaces and scattered points (denoted in the article as curveness, surfaceness and pointness) that can be retrieved. Following this idea, Lalonde *et al.* [38] classified lidar data based on their linearness, surfaceness and scatterness, and for natural terrain analysis. Figure 2.7 gives an example of what each quantity represents. In more rigorous terms, the saliency features are extracted using the theory of tensor voting, introduced by Medioni *et al.* [39]. The local point distribution is captured by the decomposition into principal components of the covariance matrix of the 3D points position. This leads to a set of three couples of eigenvalues and eigenvectors. In the case of scattered points (Pointness), no eigenvalue is predominant over the other. Indeed, no direction is ‘preferred’ by the point cloud. However, in the case of a surface (Surfaceness), one eigenvalue is significantly smaller than the other, meaning that the point cloud only have two preferred direction (e.g., the point cloud represented the ground only span in the xy plane). Finally, an edge (Curveness) only has one preferred direction, meaning that two eigenvalues are negligible compared to the third. Similarly, Heckman *et al.* [40] detected potential negative obstacles using a three-step method. They first classify the terrain using the aforementioned method of tensor voting, label the occlusions in the point cloud with ray tracing, then label negative obstacles as transitions from visible to occluded regions.

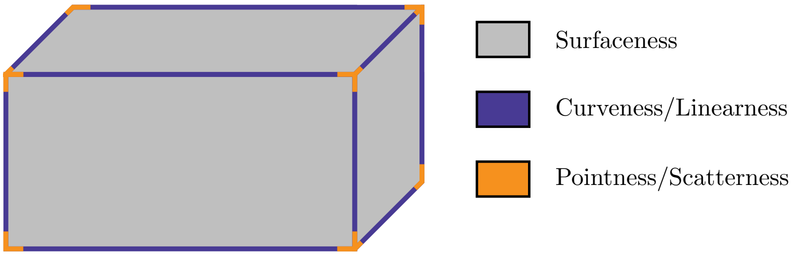


Figure 2.7: Canonic example of what surfaceness, curveness and pointness represent. In the case of a cuboid, the surfaceness represents the faces of the shape, whereas the curveness is linked to the edges and the pointness to its corners.

Following the same trend, Goldberg *et al.* [41] introduced the Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT). This system retrieves numerous information about the terrains such as step, roughness, pitch and border hazards. With these features, terrains were classified as definitely traversable, definitely not traversable and unknown by thresholding the goodness value (i.e., the suitability for travel) of each cell.

The notion of negative obstacles is also deeply investigated in the literature. Negative obstacles such as holes are often defined by the lack of data at this location. Larson *et al.* [42], [43] proposed a method based on the geometry of the 3D point cloud to classify negative obstacles. Their method, called Negative Obstacle DetectoR (NODR), classifies potential negative obstacles by detecting gaps, an absence of data, where there could exist a ditch, cliff, or negative slope. They concluded that a mixture of NODR, for long range detection, and a Support Vector Machine (SVM) classifier, for short range assessment, leads to the best results. Chen *et al.* [44] designed a method to detect traversable road regions from positive and negative obstacles, based on lidar histograms. Murarka *et al.* [45] used a stereo vision system as well as motion cues to detect both obstacles and negative obstacles (called drop-offs). Drop-offs are detected on the depth-map, where edges are classified beforehand. Comparing the motion of features above and below the edge, the system is able to infer whether the edge represents a negative obstacle. However, previous works assume that a negative obstacle is necessary lethal for the robot, even though it may be passable depending on the robot's structure and its dynamic at the time of traversal.

In that sense, some works aimed at taking into account vehicle dependant variables. One of the earliest works was done by Siméon [46] that took into account the relation between the terrain structure and its relation to the vehicle's model. Using a polyhedron approximation of the robot, contact points with the terrain were computed and each position associated with a robot configuration was labeled as admissible if it was collision-free and answered the stability conditions. The collision-free condition only states that the robot can safely rest in this pose without collision, while the stability condition states that the center of gravity of the robot must lie inside the projected polygon of the robot on the ground. Both conditions are summarized by Figure 2.8.

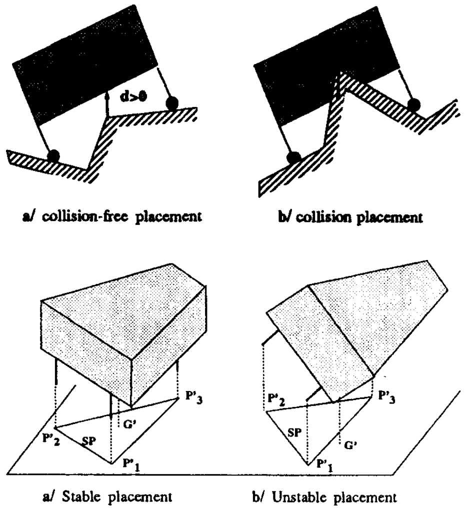


Figure 2.8: Conditions for a robot to be in an admissible state, from [46].

Furthermore, more complex models and configurations are studied in their subsequent works [47], [48]. However, most likely due to the limited computational capabilities at the time, such works did not thrive until years later. Bonnafous *et al.* [49] defined the 'danger' of a placement taking into account the angle of each axle of the robot. A placement is then said to be safe if all the angles of the axles are below a certain value, and otherwise the position is said to be dangerous. Kubota *et al.* [50] proposed an alternative approach where the robot is modeled as a cuboid, and defined a traversable area as a position where the rover can stay stably. Criteria of stability were then inferred from the robot's and terrain's geometry. Vandapel *et al.* [51] used an aerial lidar to classify the traversability of the ground. They first removed the vegetation of the scans, computing in the same time the associated confidence of the underlying reconstruction. Then, a robot footprint was superposed on the elevation map to infer its roll, pitch and ground clearance, that were after remapped between 0 (non-traversable) and 1 (traversable). Using the static performance of the vehicle, these values were thresholded and labelled as non-traversable and traversable. More recently, [52] defined a 'Dynamic Mobility Index' that explicitly takes into account the dynamic mobility of the rover. The metric is generated using dynamic simulations of the robot over various paths, that are examined through various metrics that are the stability, wheel slippage, elapsed time and energy consumption. Finally, in the context of articulated robots, Papadakis *et al.* [53] proposed a method to estimate the optimal state of the tracked robot over a 3D terrain patch. Kruijff *et al.* [54] used 3D reconstructed terrains from collapsed sites to learn and regress the mobility of the robot. Norouzi *et al.* [55] employed physics-based prediction of contact support points with the environment, using a reconfigurable robot. They used this information to plan safe paths guaranteeing the stability of the robot, taking into account its ability to change its configuration. Table 2.1 gives an overview of the presented methods based on geometric analysis for several application domains. The references are characterized by which features they take into account, namely the Terrain (Ter), the robot's attributes (Rob), the stability constraints (Stab) and the kinematic constraints (Kin).

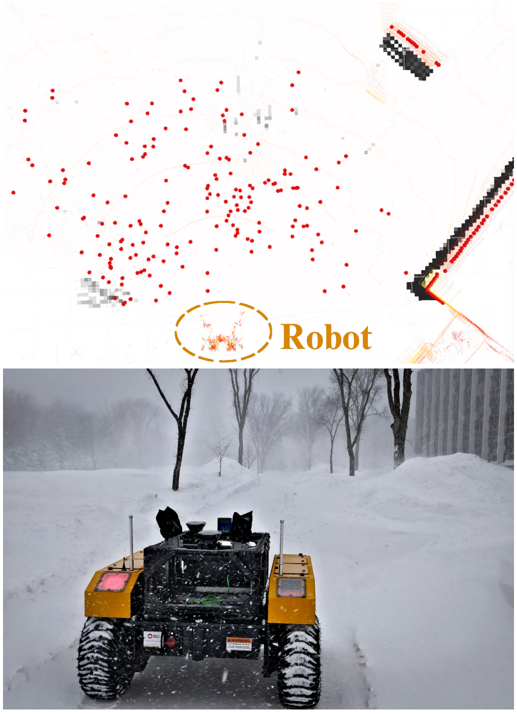


Figure 2.9: Lidar measurements (projected on the xy plane in red) and resulting occupancy grid during a heavy snowstorm in Quebec city, Canada. As easily seen, the lidar measurements are heavily impacted by the weather.

Primarily using range sensors such as lidars, such methods do not suffer from lightning conditions or shadows. Even though Papadakis [3] suggested that geometrical methods are invariant to smoke and poor weather, we argue that lidar sensors are still quite impacted by such events. Figure 2.9 shows a snapshot of an experiment conducted during a heavy snowstorm in Quebec city. A 3D lidar was mounted on the robot and a simple occupancy grid was processed based on the measurements. The measurements are massively impacted by the snow and the recurrent flurries, yielding to a very noisy occupancy grid. One can infer that geometric-based methods will have a hard time trying to process such level of noise in the environment. Another key aspect is that range sensors are still more expensive, heavier and energy-consuming than passive sensors such as cameras. These points are particularly important for intelligent vehicles and space robotics, where the payload is one of the key aspect in the specifications of the robot.

Appearance-based Traversability Analysis

Under the previous stated remarks concerning traversability analysis using geometric-based methods, another complementary aspect is the use of appearance-based methods to estimate the traversability of a given terrain. In contrast to the geometric-based methods, appearance-based methods are well suited for traversability inference of flat terrains where the robot can still get stuck, such as soft, sandy material shown on Figure 2.5 and Figure 2.10. Using only visual features, Guo *et al.* [70] proposed a method based on a SVM model that was trained to distinguish between different traversability classes using the robot odometry when crossing terrains of a given type. Andrakhanov *et al.* [69] proposed a similar pipeline, first processing the images to extract the color, texture and geometry parameters of the image and then feeding them to a deep-learning classification unit that classified the environment as traversable, conditionally traversable given the dynamic of the robot,

References	**Application**	**Criteria** Ter/Rob/Stab/Kin
Hoffman *et al.* [28]	Planetary	● / – / – / –
Siméon [46]	Planetary	● / ● / ● / –
Simeon *et al.* [47]	Planetary	● / ● / ● / –
Wright *et al.* [48]	Planetary	● / ● / ● / –
Gennery [33]	Planetary	● / – / – / –
Singh *et al.* [36]	Planetary	● / – / – / –
Stentz [56]	Planetary	● / – / – / –
Kelly [57]	Planetary	● / – / – / –
Kubota *et al.* [50]	Planetary	– / ● / – / –
Wettergreen *et al.* [58]	Planetary	● / – / – / –
Helmick *et al.* [59]	Planetary	● / ● / – / ●
Goldberg *et al.* [41]	Planetary	● / ● / – / ●
Huntsberger *et al.* [60]	Planetary	● / ● / – / ●
Ishigami *et al.* [52]	Planetary	● / ● / ● / ●
Langer *et al.* [32]	Natural	● / – / – / –
Pai *et al.* [31]	Natural	● / – / – / –
Bonnafous *et al.* [49]	Natural	● / ● / ● / ●
Vandapel *et al.* [51]	Natural	● / ● / – / –
Thrun *et al.* [61]	Desert	● / – / – / –
Lalonde *et al.* [38]	Natural	● / – / – / –
Heckman *et al.* [40]	Natural	● / – / – / –
Dubbelman *et al.* [62]	Natural	● / – / – / –
Larson *et al.* [42]	Natural	● / – / – / –
Larson *et al.* [43]	Natural	● / – / – / –
Lu *et al.* [29], [30]	Natural	● / – / – / –
Kuthirummal *et al.* [63]	Natural and Structured	● / – / – / –
Bellone *et al.* [64]	Structured	● / – / – / –
Montemerlo *et al.* [65]	Structured	● / – / – / –
Ferguson *et al.* [66]	Structured	● / – / – / –
Andersen *et al.* [35]	Structured	● / – / – / –
Ye [34]	Structured	● / – / – / –
Joho *et al.* [67]	Structured	● / – / – / –
Murarka *et al.* [45]	Structured	● / – / – / –
Molino *et al.* [68]	Search and Rescue	● / ● / – / –
Papadakis *et al.* [53]	Search and Rescue	● / ● / ● / ●
Norouzi *et al.* [55]	Search and Rescue	● / ● / ● / ●

Table 2.1: Excerpt of works tackling traversability analysis with geometric methods, adapted from [3].

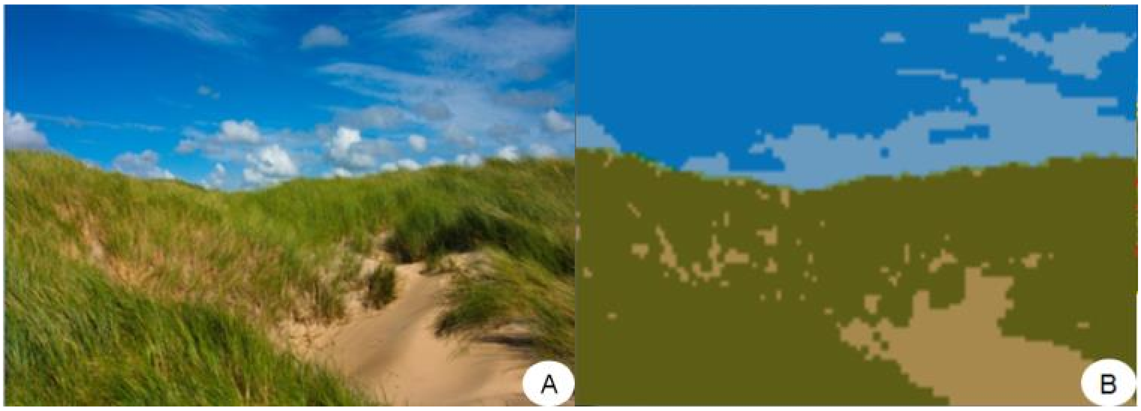


Figure 2.10: Example of terrain classification using a camera, from [69]. The image is classified here into four classes, sky, cloud, grass and sand.

or non-traversable. In the context of planetary exploration, Howard *et al.* [71] and Howard *et al.* [72] regressed the terrain traversability by measuring terrain roughness, slope, discontinuity and hardness. The roughness was computed using the size, disparity and concentration of rocky regions, whereas the slope was estimated using a neural network. Discontinuity was measured by looking at the multiple horizon lines and their distances between each other. Finally, the hardness was estimated with the textures of the environment.

In another fashion, Angelova *et al.* [23] performed a classification of the different terrain types to estimate the slippage of the robot. Kim *et al.* [73] used super-pixels, suggesting that classification at pixel-level is too prone to noise to be manageable. They managed better classification using super-pixels, resulting in more effective navigation in complex environments as shown in Figure 2.11. Alternatively, Filitchkin *et al.* [74] classified the terrain using robust features (SURF) and a SVM classifier for a legged robot. Using the classification, the quadruped robot was able to adjust its gait depending on the terrain it was crossing. Similarly, Khan *et al.* [75] used SURF features to classify outdoor terrains. Yandun Narváez *et al.* [76] used both infrared and color images to classify agricultural terrains (sand, grass, pavement, gravel & litterfall and straw-covered). Finally, Chavez-Garcia *et al.* [77] trained a convolutional neural network that inferred the traversability of aerial images, returning a traversability map that takes into account the orientation of the robot. The classifier is trained for a specific robot's locomotion type, thereby inferring more accurately the traversability of the terrain.

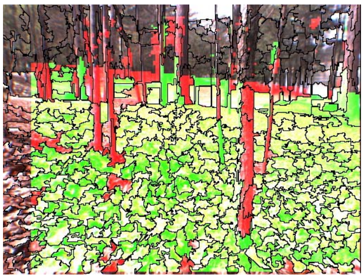


Figure 2.11: Example of traversability classification using super-pixels, from [73]. Red colored area represents non-traversable area and green colored area represents traversable area.

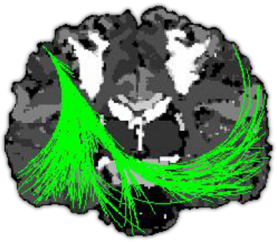


Figure 2.12: Example of path planning on Magnetic Resonance Image (MRI), from [78].

Finally, appearance-based traversability analysis can be found in other very different fields compared to mobile robotics. In the context of biomedical robotics, Caborni *et al.* [78] proposed a method for efficiently plan a steerable flexible probe for neurosurgical intervention, as shown by Figure 2.12. Using MRI, the aim was to plan for the least risk path (i.e., the path that yields the better chance of survival and the least sequela for the patient). Vaillant *et al.* [79] introduced an equation for cost calculation on a path in a brain, that is very similar to the ones used in mobile robotics.

In overall, appearance-based methods are a well suited answer for the aforementioned problems of geometric-based methods, at the cost of being dependent on the lighting conditions. An overview of the presented appearance-based methods can be found in Table 2.2. Using conjointly both approaches, when possible, will yield the best results available.

References	**Application**	**Criteria** Ter/Rob/Stab/Kin
Howard *et al.* [71]	Planetary	●/ – / – /–
Howard *et al.* [72]	Planetary	●/ – / – /–
Angelova *et al.* [23]	Natural	●/ – / – /–
Kim *et al.* [73]	Natural	●/ – / – /–
Guo *et al.* [70]	Natural	●/ ● / – /–
Andrakhanov *et al.* [69]	Natural	●/ – / – /–
Chavez-Garcia *et al.* [77]	Natural	●/ ● / – /–
Yandun Narváez *et al.* [76]	Natural	●/ – / – /–
Khan *et al.* [75]	Natural and structured	●/ – / – /–
Filitchkin *et al.* [74]	Natural and structured	●/ – / – /–
Caborni *et al.* [78]	Neurosurgery	●/ – / – /–
Vaillant *et al.* [79]	Neurosurgery	●/ – / – /–

Table 2.2: Excerpt of works tackling traversability analysis with appearance-based methods, adapted from [3].

2.2.2 Risk As a Collision Analysis

As seen in the previous section, mobile robots already have a hard time assessing risk in a traversability-based approach. This method mainly deals with the wheel-ground interactions, in order to avoid the robot of getting stuck in loose soil. However, a more intuitive risk is also to take into account as soon as our robot will take the road (or the planet, depending on the application): is the path *collision-free*? As opposed to the traversability approaches, we will here assume that the robot can only be in one of two states. The robot can either be free of collision, therefore safe (i.e., the risk is null), or either currently collides with an obstacle such as a boulder, a vehicle or a pedestrian, and therefore not being safe (i.e., the risk is not null).

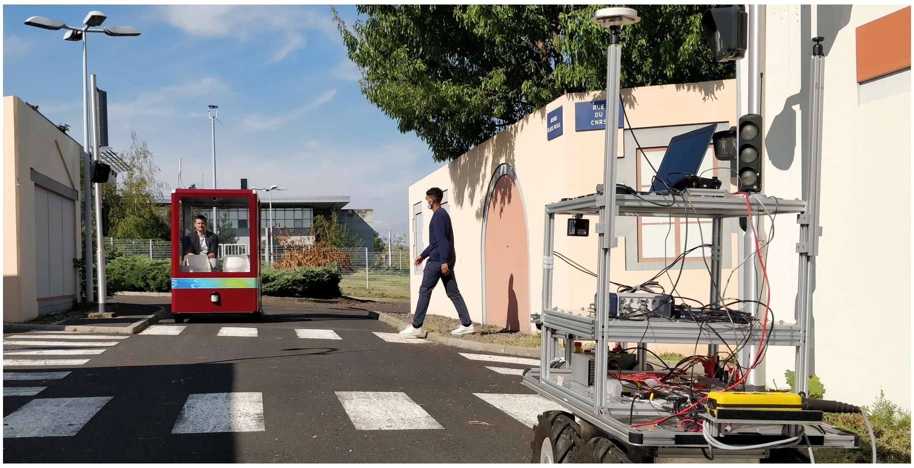


Figure 2.13: Example of scenario a mobile robot might encounter, from [80]. In this case, a 'risky' event is the robot colliding with either the pedestrian or the other vehicle.

Without any doubt, collision-based risk assessment has been the risk the most studied in the context of intelligent vehicles. Indeed, in an urban environment, the main concern is to avoid at all cost any collision with the environment but also with the other agents. Figure 2.13 shows an example of such scenario. The robot, in the right of the picture, is currently going through a crossroad with multiple other agents. Another vehicle is coming on the other lane, while a pedestrian is crossing the street. As such, any decision of the robot has to take into account the possibility that it will result in a collision with either the environment or one of the agents.

But first, why do we assume that a collision is risky? While we will deeply discuss this matter in Section 2.3, it is interesting to find a good intuition in the case of collision-based risks. Collisions, in themselves, are not *bad*; their consequences, however, are often undesirable. Let us consider the following example depicted in Figure 2.13 where a pedestrian is crossing the road, and assume that the robot does not have the time to stop. In this scenario, the robot has three choices which are

1. Brake and collide with a wall, if possible;
2. Brake and collide with the pedestrian; and
3. Brake, swerve and collide with the vehicle on the other lane.

For each choice, we may ask ourself what is the associated risk. For the first one, the risk is indubitably the damages to the vehicles induced by the collision. However, in the two other cases, another risk is present, which is the damages done to the collided agents. In that sense, we may prefer to collide with a wall even if it is, for the vehicle, more dangerous than collide with the pedestrian.

Collision-based risk assessment also finds its roots in other fields. For instance, Haddadin *et al.* [81] studied extensively the injuries from robotics arms. Indeed, manipulators can not only cause blunt trauma but also contusions, stab/puncture wounds, abrasions or lacerations. Although interesting, we will limit this section to the field of intelligent vehicles.

In this section, we investigate how the related works choose to define and manage collision-based risks. We split the section into three main parts, namely the deterministic indicators, the probabilistic ones, and the new trend in robotics that is the learning approaches.

Deterministic approaches

Historically speaking, the deterministic approaches were the first to be developed. Many indicators have emerged, taking into account different types of criticality. In that sense, Fraichard *et al.* [82] and Fraichard [83] proposed a framework to guaranty that the robot will not create any collision, using the concept of 'Inevitable Collision State'. The aim of the robot is to always avoid such state where a collision becomes inevitable. This theory provides safety criteria, saying that the robot should always consider its own dynamics, the environment object's future behavior and reason over a quasi-infinite time horizon. However, such criteria are hard to entirely respect in complex environments such as the open road. Furthermore, the research does not address what to do if the robot is in an inevitable collision state: should the robot brake, steer, or engage in an evasive maneuver? Thus, more fitted metrics have been developed in parallel for intelligent vehicles where accident might always happen.

In the context of intelligent vehicles, the first and most simple risk is the change of velocity between the vehicles due to the collision. Brännström *et al.* [84] stated that in case of inevitable collision, the system has to brake as much as possible to limit the velocity at the time of collision. Although such metric seems too simple to effectively work in every scenario, it can yield more complex metrics that cover more situations. Labayrade *et al.* [85] proposed a collision mitigation system whose aim is to decrease the kinetic energy dissipated during a collision through emergency braking. In the case where the collision between the vehicle and the obstacle is inevitable (and the vehicle does not have a choice of which obstacle it collides with), decreasing the dissipated kinetic energy reduces to brake as much as possible before the impact but also choosing the right angle of collision. Various works [86]–[89] chose to model the vehicles as geometrical shapes such as ellipses, circles or polygons. The risk was then defined as the amount of overlap between the shapes. Using generic motion-based object detection, Keller *et al.* [90] proposed a method to mitigate collisions with pedestrian by either braking or steering. Noh *et al.* [91] assessed collision risk on highway in order to take the best course of actions.

Despite the growing number of risk metrics for intelligent vehicles, the Time To Collision (TTC) continues to be the most popular. Introduced by Hayward [92], the TTC measures the remaining time the driver has left before a collision happens, assuming constant velocity. This metric, despite its age, remains the most used in the domain of intelligent vehicles. This is due to its simplicity, low computational time and is easily improvable. Table 2.3 gives some examples of TTC metrics developed over the years. However, as pointed by Laugier *et al.* [93], the TTC metric suffers from its lack of context and becomes less and less effective as the time horizon increases. Figure 2.14 shows some examples where the TTC can underestimate and overestimate the risk. In the first example of a crossroad where all the obstacles are stopped, the TTC will be set to infinite, therefore meaning that the situation in one hundred percent safe. However, such scenarios are one of the most hazardous situations as many accidents happen in intersections. In the second example, because of the road's curvature, the TTC results in a high risk even though the situation is not risky.

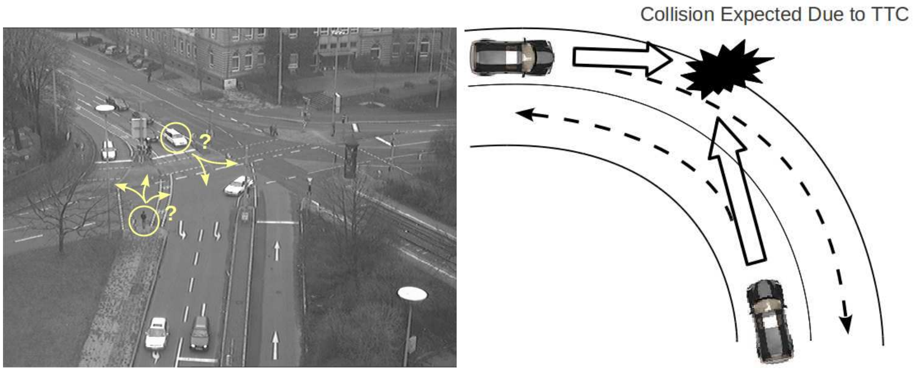


Figure 2.14: Limitations of the TTC metric, from Laugier *et al.* [93]. Because it lacks context, the TTC can underestimate the risk at intersection (left figure) or overestimate it in simple scenarios such as a road turn (right figure).

In parallel, other metrics have been developed for risk assessment in road scenarios. Allen *et al.* [98] proposed the Post-Encroachment Time (PET) metric, which is the time between the moment that an obstacle last occupied a position and the moment

Approach	**Description**	**Contribution**
TTC [92], [94]	Definition of the TTC	Introduction of the metric [92] whereas Lee [94] demonstrated that the TTC is an easy metric to be picked up by the driver and is sufficient for collision mitigations.
Extended TTC [95]	Extend the TTC indicator	Improved safety indicators using vehicles trajectories, used to counter the TTC that may vary too much over time.
Improved TTC for imminent danger [87]	Takes into account the acceleration, lane-changing maneuvers and trajectories that are not constrained by lane-based concepts	The focus is to propose a metric that is not just an 'expected or actual time-to-collision' but a metric that measures the imminent danger faced by the driver (the possibility that an accident will happen) if he/she continues on the current trajectory
More accurate TTC [96]	Improve the TTC calculations between two vehicles colliding at constant speed along a straight path	Improve the TTC calculations with a more accurate method taking into account the geometry of the vehicles and remove a dependency to a parameter that was hard to tune.
TTC for unconstrained vehicle motion [97]	Generalize the TTC metric for 2D environments	Generalize the TTC calculations to unconstrained vehicles motion, therefore working for all general traffic scenarios.

Table 2.3: Some examples of TTC measures

where the robot reaches the same position. This metric is particularly interesting for highway situations but does not translate well to more complex environments. In the same context, Hupfer [99] introduced the Deceleration to Safety Time (DST) metric, processing the required deceleration that has to be applied to the vehicle to maintain a given safety time with the other vehicles. Hillenbrand *et al.* [100] used the Time To React (TTR) metric to calculate the remaining time to avoid an imminent collision by emergency braking, steering, or a 'kick-down' maneuver such as leaving the collision zone early enough.

In a more empirical manner, Richards [101] published a study relating the relationship between the speed of the vehicles and the risks of fatal injuries for pedestrians and car occupants. For instance, Figure 2.15 shows the relation between the impact speed and the severity of the injuries for frontal collisions. Such metrics, at the cost of relying on real accident data, can offer more nuanced approaches as they do not rely on parameters and thus yield more meaningful decisions.

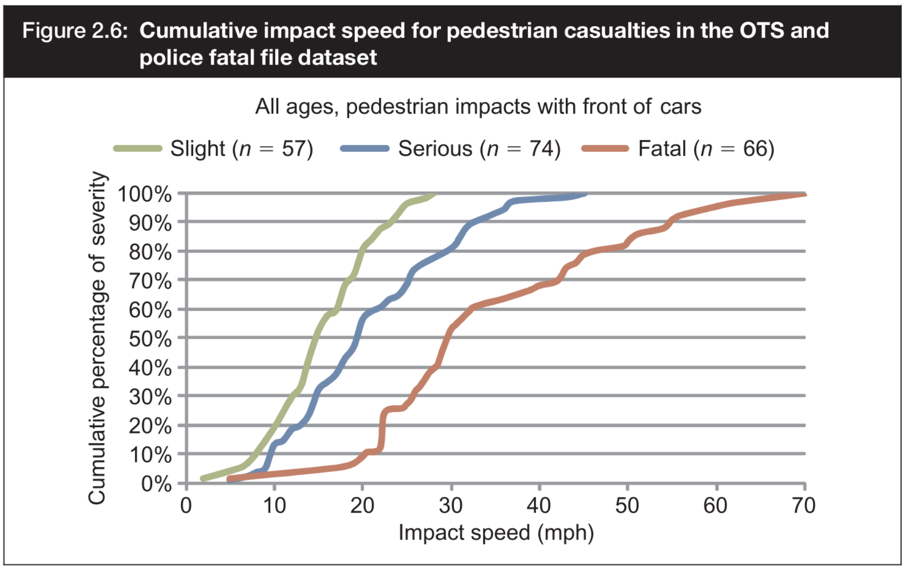


Figure 2.15: Relation between the impact speed of the vehicle and the severity of the collision for the pedestrian, from [101].

Probabilistic approaches

Subsequently, the deterministic approaches started to show their limitations. Indeed, their main asset is their low complexity and therefore low computational time. However, such strength somewhat fades away as computer's capabilities grow. These methods are not able to deal with the unpredictable. Following the large trend of the probabilistic robotics [1], the aforementioned methods were extended to the probabilistic realm, of which we give here an overview here.

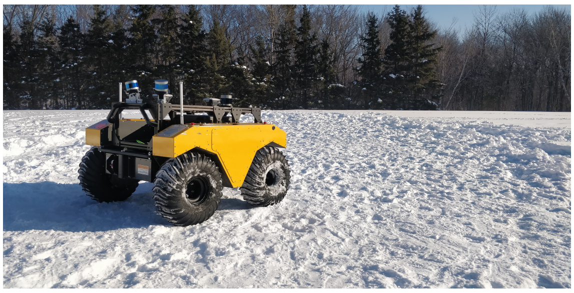


Figure 2.16: Example of terrain where deterministic prediction of the robot's displacement would fail: because of the snow, the robot's skidding and slipping are almost impossible to estimate accurately.

The uncertainty can be taken into account in multiple ways. At first, one can model the uncertainty of the robot's displacements. Indeed, in every collision-based risk assessment system, a prediction step is involved. In contrast to traversability based systems where the robot can assess the traversability by safely crossing it (e.g., at low speed), a collision-based system aims to avoid all the collisions. Therefore, the systems need to predict the robot behavior and assess the confidence on each command. As an example, let us imagine a robot as depicted on Figure 2.16 evolving in a sub-arctic environment. Because of the deep snow of uneven density, the robot's displacements are hindered and heavy slippage and skidding make any accurate estimation of the displacements impossible. As such, any deterministic method would only take into account the most probable motion, even if there are little chances that its actual displacement is near the estimation. These problems are specially critical in the case where the robot has to negotiate sharp turn or plan in a crowded environment, i.e., where the robot has little room for maneuvers. In the context of intelligent vehicles, the same problem arises when, for instance, snow or heavy rain is falling on the road. Mesbah [102] wrote a survey of stochastic methods for Model Predictive Control (MPC), allowing robots to plan while taking into account its future probabilistic motions.

On the other hand, the decisions of the other agents in the environment are almost always unknown by the robot. Henceforth, it becomes critical to reason over the possible future movements of the agents instead of only the most probable one, often assuming that the obstacles keep their velocity and direction constant over the risk assessment horizon. This assumption is not necessarily bad but deteriorates for longer time horizons. For instance, short TTC metrics using this assumption are close to the reality and yield correct decisions for accidents mitigation. However, in more complex scenarios where the aim is to avoid entering an accident scenario (e.g., adapting its speed before crossing an intersection), context and hypothesis have be taken into account. As such, probabilistic methods have been developed to tackle these cases.

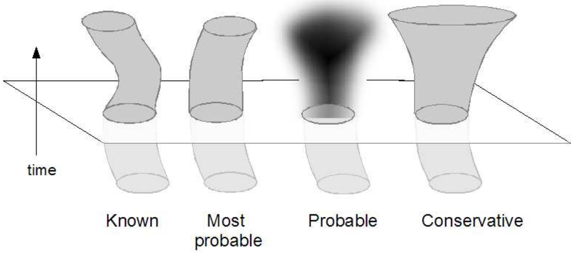


Figure 2.17: Example of conservative approach from [103], where the region below the plane was observed and the region on top is predicted. For a known trajectory (on the far left), the most probable prediction is not enough to provide safety guaranties. Instead, a conservative approach thresholds the probability distribution, yielding to a wider prediction but that is almost sure to contain the real future trajectory.

It is worth noting that dealing with probabilistic obstacle often lead to a far greater computational cost and complexity in the theory. As such, many works aim at dealing with these cases by using a conservative approach that remove the complexity of the probabilistic approaches and fall back on deterministic works. Benenson *et al.* [103] extended the work of Fraichard [83] to take into account that sensors only yield partial and uncertain information about the environment. As dealing with probabilistic obstacles prevent the framework from generating safety guaranties, they proposed to use a conservative world model by thresholding the probability distribution, as shown in Figure 2.17. Using this thresholding method, the framework can deal with deterministic obstacles and guaranty the safety of the robot under the condition that the probability of the obstacles being outside the conservative set is negligible. Finally, they use a 'harm value' to characterize the environment at each point. This value can take into account numerous factors such as the occupancy, the type of obstacles, and so on. A path is said to be safe only if its harm value is zero for every position of the path. Bouraine *et al.* [104] extended the aforementioned work with a more tight conservative approach and the introduction of Braking Inevitable Collision State that allows robots to evolve in a limited field-of-view environment.

In another fashion, [105] proposed a risk mitigation framework that assigns a Probability of Collision with Injury Risk (PCIR) value to each position of the environment. Contrary to [103], this risk takes into account that the higher the speed of the vehicle is, the deadlier is a collision. They propose to weight the injuries (e.g., injury to the vehicle, to the collided pedestrian, and so on) and therefore leave the risk definition generic for each application. We will come back to this very interesting notion in the next chapters. Sandblom *et al.* [106] modelled the dynamic obstacles as probabilistic entities for autonomous braking in case of collision. Likewise, Althoff *et al.* [107] considered uncertainties coming from the measurements and the possible behaviors of other traffic participants, yielding a probability of crash for a specific trajectory.

Berthelot *et al.* [108] extended the TTC metric to the probabilistic domain using states estimations of the surrounding vehicles. In the same way, Eggert [109] proposed a generalization of the TTC metric. They used the TTC to define a risk as the expectancy of the damages to the car over the prediction time.

Fulgenzi *et al.* [110] presented a navigation algorithm that detects and tracks obstacles on-line. Their contribution is to assume typical motion patterns represented by a Hidden Markov Model (HMM). Wang *et al.* [111] defined the risk as the probability of collision, but used two methods to compute it: the first indicator shows how probable a collision will happen in the near future, whereas a second long-term risk indicator informs of a collision possibility over the whole time horizon. In the same context, Oboril *et al.* [112] proposed to model the risk as the expectation of the collision severity. They modeled the collision severity using an inelastic collision model but argued that more sophisticated ones can be used if further information is available.

Finally, one can provide some examples in personal assistance robotics. Rios-Martinez *et al.* [113] proposed to add to the standard risk of collision the risk of disturbance, defined as breaking social rules such as crossing between two people talking. Ogorodnikova [114] defined the risk as a sum taking into account among others the likelihood of avoidance, the severity of the collision and human interactions.

Learning approaches

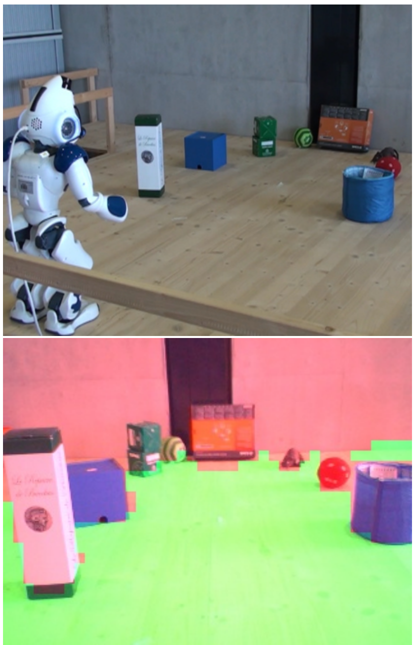


Figure 2.18: Example of learned traversability from [115], where collision-free zones are in green, otherwise red.

Since the last decade, the explosion in computational performances of the machines allowed the learning methods to get a second wind and thrive in many robotics applications. Without implying that non-learning approaches will be outperformed by the latter, such approaches yield interesting results in a great number of domains [116]. Maier *et al.* [115] proposed to learn collision free zones in an image using a sparse range sensor on the head of the robot, as shown in Figure 2.18. Indeed, because of the position of the range sensor, the robot would have to tilt his head numerous times to assess the risk of the entire environment. They taught the robot to learn from the range measurements, identifying the traversable areas (in the sense of collision-free zone) with only a camera. In the same fashion, Kostavelis *et al.* [117] used SVM classifiers to separate the traversable and non-traversable environments with a stereo vision sensor. Heo *et al.* [118]

presented a method to learn and detect collisions in the context of industrial robotics: as human-robot interactions tend to increase over the years, safe but also reliable framework for collision detection are needed. A neural network is trained to learn collision signals while limiting the false negatives, leading to safe and efficient interactions. Learning approaches have the quality to better generalize than standard approaches, with the drawback of being very little explainable and yielding no guaranties of the framework output. Furthermore, large datasets have to be used, whose constructions can be tedious. Koert *et al.* [119] taught a robotic arm to grasp and plan collision-free trajectories, while generalizing enough to be able to adapt in other environments than the one they were taught in. Malm *et al.* [120] takes the discussion further by suggesting that detecting a collision is not sufficient for safe robotics. As an example, a robotics arm can stuck an operator between itself and the wall, therefore detecting a collision and stopping. However, in this configuration, stopping is not the best course of action since the robot is still applying pressure over the operator and the wall, harming the human. The robot would better choose to detect the collision and move the arm elsewhere (without of course colliding anything else). This study embeds a notion that is not very much discussed in the robotics community, that is the *post-collision* risk assessment. Most works stop at avoiding a collision or minimizing the collision if inevitable, but it is also important to look at what happens after (e.g., colliding a pedestrian then a wall is far more dangerous for the pedestrian than if the car just stopped right after the collision). However, dealing with such cases add another layer of complexity while there is not yet a consensus on how to deal with the risk in *pre-collision* scenarios.

2.2.3 Risk As an Efficiency Analysis

In the last two sections, we looked at the risk from two very different perspectives. The first one focuses on the interaction between the robot and the ground to assess if a robot is able to enter and leave it without harm. The second one targets the collisions with the robot and the environment, taking into account that neither the robot nor the environment is invincible and therefore encounters at high velocity can cause harm. In this section, we discuss and investigate one last form of risk, that is the risk coming from efficiency. Although less common, it remains one of the key aspect to take into account in many applications.

Let us consider some examples to better grasp this concept. For a rover on Mars, energy efficient planning is mandatory. As such, it can be very risky to send the robot on a long trajectory that might deplete its energy reserves and leave it in a tedious situation such as a sandstorm that will cover its solar panels and hamper it from recharging its batteries. In the context of search and rescue robotics, efficiency is the first goal of the robot. Its aim is to find as many as possible endangered people even if it means endangered itself if it does not stop him from continuing its mission (i.e., finding more people). The main risk can be seen as an efficiency problem, where the robot has to be as fast as possible. Of course, traversability and collision based risks have to be considered since crashing means failing to find other people.

Some authors chose to oppose the notion of efficiency and risk [121], since collision and traversability risks are often inversely affected by the speed. Feyzabadi *et al.* [122] proposed to separate the efficiency from the risk in an industrial application by maximizing the efficiency under the constraint that the risk is bounded to a used-defined value. In this framework, the risk is generic and take into account both the position in the environment but also the action the robot underwent to reach this location.

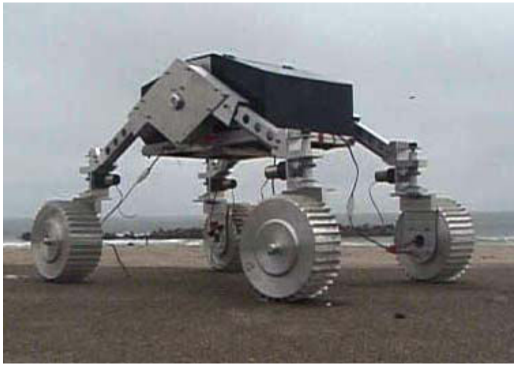


Figure 2.19: Rover used in the experimentation from [52]. In this scenario, the risk of running low on energy is as risky as entering a low traversable area.

On the opposite, Ishigami *et al.* [52] and Martin *et al.* [123] take into account the energy consumption as one of the traversability factor for the robot. In [52], the path evaluation takes into account both the energy consumption and elapsed time, among traversability-based metrics. As such, the rover (shown in Figure 2.19) is able to chose more efficient paths that can be less traversable. Martin *et al.* [123] constructed large scale traversability maps taking into account the slip but also the power consumption, allowing the robot to plan paths more efficiently in the environment. Finally, Sakayori *et al.* [124] created an energy-efficient trajectory planning method for a wheeled robot in slope ascending scenarios. Using extensive simulations, the robot estimates the power consumption for a given slope and approximates it with a neural network. Working with the calculated model, the robot is then able to generate efficient trajectories to climb hills.

2.3 What is a Good Risk Metric?

In the last section, we presented the main trends in risk assessment. However, as one can easily notice, many definitions exist and may not equivalently quantify the true danger of a situation. In parallel to the more applied aforementioned researches, some authors tried to bring together the definitions, proposing a unifying framework.

First, we present a short discussion on the more philosophical aspects of the risk. Then, we discuss the main trends of the unification of the risk definitions.

2.3.1 The contradiction of the utilitarianism

The utilitarianism is an ethical theory based on the maximisation of the happiness and well-being for all affected individuals. This ethical position emerged in the 18th century and has been first formulated by Jeremy Bentham. In [125], he defined a novel definition of moral: any action is measured by the amount of good it does to the world. This theory is classified as consequentialist, where each action is not good or bad by itself but is defined as its consequences on the world (i.e., the individuals). Therefore, it strongly opposes deontological morals, such as the one developed by Emanuel Kant, where an action is intrinsically good or bad. Over the years, the utilitarianism evolved toward a more complex moral. Whereas Bentham was only considering instant gratifications, its disciple, John Stuart Mill, proposed to replace it with a broader notion that he called happiness [126]. The happiness distinguishes several types of pleasures where some are more valuable than others (e.g., long term health is more valuable than a short pleasure such as eating a cake). The main appeal of the utilitarianism in the case of risk management is its mathematical formulation. Indeed, the utilitarianism wants to maximize the happiness of the world. As such, one does need a metric to evaluate this happiness, and therefore yield a metric that is able to evaluate the change of happiness any action would yield. Of course, doing such computation is realistically impossible: however, a robot might be able to approximate it in a given scenario, assuming rightfully that its actions will not change the state of the world outside a given region around it.

Using the theory of the utilitarianism, the risk is simply the amount of happiness the world would lose compared to the best decision the robot might take (i.e., the one maximizing the happiness). As shown by Karnouskos [127], in the hypothetical scenario where an intelligent vehicle has only two choices, either colliding with a group of pedestrian or throwing itself of a cliff, hence killing the one person in the car, the utilitarian choice would be to kill the occupant of the car instead of the pedestrians. At this point, one can ask himself if, at first, there truly is a better solution over the other. Indeed, the opposite point of view also makes sense. As called in [127] as the 'self-safety' approach, it aims at always thinking first of the safety of its occupants. In overall, this opposition between the two approaches is paradoxically reflected by the public mass: although more people would like to see utilitarian cars on the street, they are less prone to buy a self-driving car if it is configured to prioritize the safety of the pedestrians over the occupant.

Indeed, the problem is far more complex than a simple utilitarian calculation or a deontological approach: as said by Ernst Tugendhat, a doctor willing to save five patients does not have the right to kill a perfectly healthy person for its organs, even though the utilitarian cost would dictate to do so. This problem can be translated into a driving scenario where some pedestrians cross the street even though the crosswalk light was red.

This discussion could last for an entire book, but we can easily see that it seems that no risk calculation will easily surpass the others in every situation. Therefore, the risk in robotics in often simplified to avoid collisions or not be the one to cause them[1], as proposed by Fraichard [83] and Shalev-Shwartz *et al.* [128]. Other works, like ours **Laconte2020**, [80], [129], choose to keep a utilitarian approach, minimizing the overall damages in case of an inevitable accident. Furthermore, as robots can only estimate the state of the world, the decisions often involve probabilistic movements. Majumdar *et al.* [130] proposed a formalization on how a robot should distill the probabilistic estimation into a single number that is the risk or the utilitarian cost of an action.

2.3.2 Unification frameworks

In this section, we investigate the main trends that tried to unify and propose a generic framework for risk management.

Inevitable state collision

In his work, Fraichard *et al.* [82] defined the notion of inevitable collision state: a robot is said to be safe if it never enters such state, defined as a state where a collision in the near future is impossible to avoid. This notion is extended in [83] where the author proposed safety criteria that are meant to better understand key aspects relative to the safety issue. The criteria are

1. to decide its future motion, a robotic system should consider its own dynamics;

---

[1] Hence leaving the utilitarian dilemma to the others agents.

2. to decide its future motion, a robotic system should consider the environment objects' future behaviour; and
3. to decide its future motion, a robotic system should reason over an infinite time-horizon.

The third point can be relaxed to the extent that the time horizon should contain the time required to reach the goal state. Then, it is shown that the inevitable collision state framework does take into account the three criteria, therefore yielding a conservative, safe framework.

Figure 2.20: Example of small robot (Roomba) that is not harmful even in case of collision with a pet or furniture.

In the same paper [83], the authors discussed whether safe robotics systems exist in 2007. The emphasis is on 'harmful' robots defined here as robots whose size and dynamics make them potentially harmful for themselves or their environment. Small robots such as a Roomba shown in Figure 2.20 are not harmful in this definition and do not have to take into account the above criteria, as bumping into a pet or furniture is, citing [83], 'no big deal'. Although the authors chose to completely separate 'harmful' robots from 'harmless' ones, this consideration will be the premises of more complex risk metrics. Indeed, the risk is therefore intrinsically linked to the force of collision, itself dependent on the mass of the robot. In the case of harmful robots, even though lots of robots have been successfully evolving in crowded environments such as a warehouse or a museum, it has been shown that each system does violate at least one of the above safety criteria. Although effectively avoiding collision, one can ask which agent is truly responsible for avoiding the collisions. Indeed, it is suggested that instead of the robot perfectly avoiding the pedestrian, it is actually the other agents that took care of the collision avoidance. As such, putting the robot in an environment with blind people would lead poor results.

Taking into account the remark that dynamic obstacles coupled with limited field of view yield to numerous problems and prevent formal guaranties of safety, Bouraine *et al.* [104], [131] relaxed the notion of inevitable collision state to the passive motion safety. This weaker level of safety states that, in case where a collision must happen, the robot will be at rest. It echoes to the last section discussion, where the robot will here leave the risk (i.e., utilitarian) calculation and management to the other agents, namely the crowd or the other robots. As such, the aim is to avoid at all cost a state called braking inevitable collision state, that is a state where the robot is not able to stop before the collision. The authors argued that passive motion safety can be too limited, and more sophisticated levels of motion safety could be explored, such as the passive friendly motion safety introduced by Macek *et al.* [132]. This safety requirement guaranties that if a collision is inevitable from the robot's point of view, the robot will be at rest and the other actor in the collision will have the time to either stop or avoid the collision.

Alternatively, Benenson *et al.* [103] defined the notion of safe motion: a robot is said to be safe if it can be guaranteed that its motion will not harm himself or its surroundings. Once again, it is necessary to explicitly define what is behind the word 'harm': here, a robot can harm itself by traversing inadequate terrains (e.g., water, negative obstacles) or pushing itself out of its range of dynamic stability, while it can harm its surrounding by colliding with it. Compared to the previous framework, it also takes into account a more traversability-based risk management, as talked in subsection 2.2.1.

Safety validation of automated driving

In the specific case of self-driving vehicles, robots have already started to cohabit with human drivers. As such, several approaches exist to validate the safety of the methods that are implemented in these vehicles. Safety validation is often understood in the sense that a mathematical proof can be given. However, such an assertion can be extremely hard to obtain. Indeed, compared to industrial robots where very few agents are around the robot and the environment is fairy well controlled, intelligent vehicles are expected to meet numerous different actors and situations.

Sivak *et al.* [133] wrote a survey of the general limitations and safety challenges in the context of self driving vehicles. The first and not the least is the aforementioned problem of road sharing. Not all crashes are caused by the driver (AI or human). Some accidents are direct causes of other road participants such as pedestrians that jump on the road, dysfunctional brakes, environmental hazards (e.g., potholes or dense fog). As provided in [133], a simple example of a drunk pedestrian stepping unexpectedly on the road is a perfect example showing that perfect safety does not exist. Because of the short distance between the car and the pedestrian, there might not be the time to stop or at least divert the trajectory enough to avoid him. However, an artificial intelligence would have a quicker response time than a human driver and therefore can brake sooner, leading to a smaller injury for the pedestrian. Figure 2.21 gives another example of a vehicle that cannot guaranty its own safety, as any other agent can choose to collide with the robot. The second point is the vehicular factors: some crashes are caused by vehicle failures, such as dysfunctional brakes or sensors failures. In overall, [133] argued that given the complexity of the hardware needed for autonomous driving, vehicular failures will occur more frequently compared to conventional vehicles. Finally, environmental factors also hinder the road safety. Even if such cases are more and more tackled successfully, potholes, debris or sudden weather changes still represent a challenge and current sensors and framework might not be able to take them entirely into account.

Given these considerations, we conclude that the issue of safety validation cannot be 'solved', as in proven mathematically with absolute certainty, at least without assuming numerous conditions such as sun illumination, other agents, perfect road and so on. Junietz *et al.* [134] and Serban *et al.* [135] provide an evaluation of the current safety validation methods. As stated in [135], the current ISO standard for intelligent vehicles, ISO 26262, fails to cover emergent concerns related to autonomous decisions (i.e., path planning). Indeed, the ISO 26262 standard only provides a way to manage functional safety thorough the system and every subsystem, but does not concern decision making. As such, Serban *et al.* [135] proposed a first step toward a safety analysis for autonomous behavior, based on the ISO norm. Further research is however needed to converge to a usable framework.

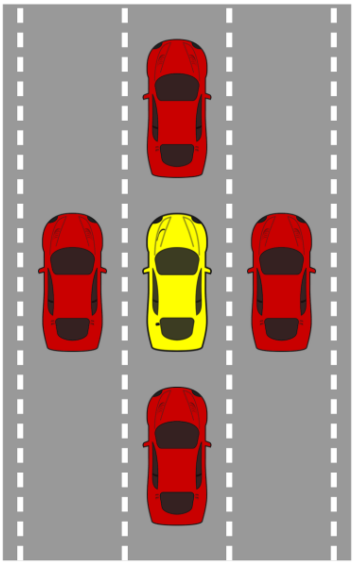


Figure 2.21: Example of scenario from [128] where the autonomous vehicle (in yellow) cannot guaranty its own safety, since any other agent (in red) can crash into it without the autonomous vehicle being able to evade it.

Finally, we present one formal model that is currently employed in many autonomous vehicles, that is the Responsibility-Sensitive Safety (RSS) from Mobileye. In [128], they proposed a method to formalize and standardize safety assurance that is scalable to production and commercialization. As argued in [135], they state that system safety is necessary but not sufficient. Indeed, a bug-free autonomous vehicle can perfectly crash in every pedestrian it sees. They propose a system called RSS that is constructed around five rules: 1) Do not hit someone from behind; 2) Do not cut-in recklessly; 3) Right-of-way is given, not taken; 4) Be careful of areas with limited visibility; and 5) If you can avoid an accident without causing another one, you must do it. Then, each rule is formalized and integrated into the framework. However, such a method assumes perfect detection of the other agents, thing that is still nowadays quite challenging.

Quantification of a stochastic risk

In contrast to the last section, the following methods do not require a perfect knowledge of the world. Instead of dealing with known obstacles, they deal with probabilistic environments. First, one can assume a finite world, i.e., a world defined by a finite set of states and actions that allow transition between the states. A simple example would be a domestic robot charged to water the plant: the states of its world are $\{\text{watered}, \text{dry}\}$ and the only action is watering the plant. Figure 2.22 shows the graphical representation of such worlds. Each state evolves using an action from the robot, where the states are composed of characteristics (e.g., is the plant watered or the ambient temperature).

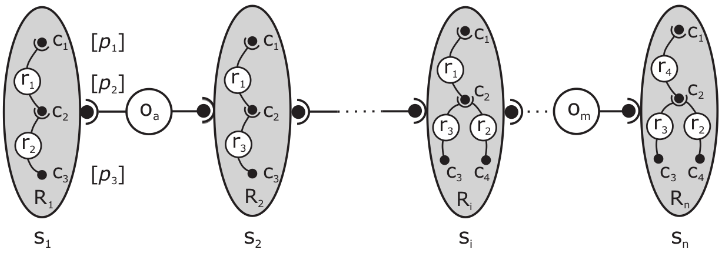


Figure 2.22: Example of environment representation in a finite world, from [136]. Each state s_i is composed by several characteristics c_j that are linked by relations r_k. The world change state using operators (i.e., actions) o_x.

Using this representation, Ertle *et al.* [136] quantify the risk of an action the robot might undergo using a triplet that is 1) what can happen with this action; 2) how likely is that; and 3) what are the consequences. In our toy example where the robot has to water a plant, one can think of the risk to miss the plant and water a power outlet, for instance. Using the above formalism, the triplet would be 1) water+electricity; 2) Probability to miss the plant pot and touch a power outlet; and 3) Probability of injury. Although interesting, such method requires high-level information about the world and does not yield a risk metric but a triplet that is harder to use in, for instance, path planning algorithm.

As such, Majumdar *et al.* [130] provided a comprehensive study of risk quantification. As discussed earlier, any artificial intelligence needs to assign a cost to each action, that is a mapping from a change in the environment caused by the action to a real number. Section 2.2 gives numerous examples of such mapping, including the well-known probability of collision for intelligent vehicles. Before this amount of different metrics, Majumdar *et al.* [130] tried to answer the question of what constitute a ‘good’ risk metric. In order to provide such answer, they propose several axioms that should have a risk metric. They illustrate their axioms using a risk that is the monetary cost of each action (e.g., an accident is costly, colliding with a pedestrian is even more expensive, whereas driving without accident only costs the price of the gas). Then, they demonstrate that some metrics, although quite popular, do not fulfill the axioms and therefore lead to unreasonable choices. As an example, for an action that leads to the event C (e.g., the force of collision induced by the action, where 0 indicates that no collision happened), one has different ways to convert the probability distribution to a real number. Figure 2.23 shows the most popular methods to quantify a probabilistic risk. The expected value as well as the worst case constitute simple

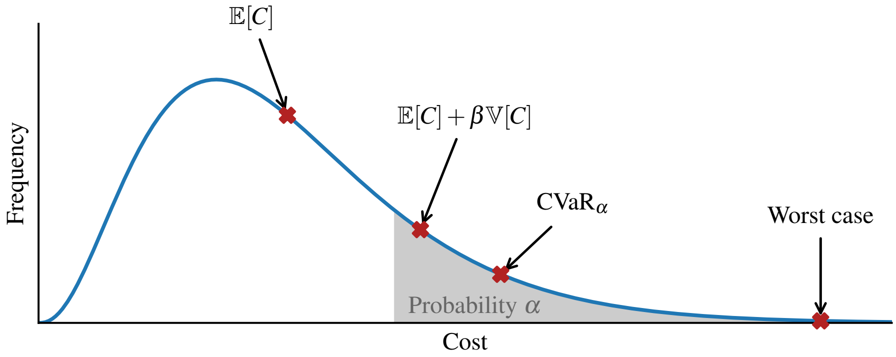


Figure 2.23: Illustration of the most used risk metrics. From left to right: Expected cost, Mean-Variance, Conditional Value at Risk, and Worst case.

and reasonable metrics, in the sense that they satisfy all the axioms. However, they demonstrate that the well known 'Mean-Variance' metric, that sums the mean of the distribution and the variance times a real factor, does not satisfy the axioms and thus leads to inconsistent risk metrics in certain situations. As such, this metric should not be used. Finally, a more complex metric is the Conditional Value at Risk (CVaR) and gives another point of view of a distribution. Whereas the expected value answers the question 'What should I expect from this action?' and the worst case obviously answers 'What is the worst thing that could happen?', the CVaR answers the most complex question 'If something goes bad, how bad is it?'. More mathematically speaking, the CVaR corresponds to the expected value given that the random variable is in the long tail of probability α (gray area in Figure 2.23). Using the theory and the axioms developed in [130], one can easily choose a metric that will not lead inconsistent results once deployed in the field.

As an example of risk assessment using CVaR, Hakobyan *et al.* [137] proposed a method to navigate between randomly moving obstacles. Figure 2.24 illustrates the problem: given a list of obstacles randomly moving in the environment, the robot needs to find a way to reach its goal while taking the minimum possible risk. To do so, they formulated the problem as a random variable which represents the distance the robot needs to travel to reach a safe region (i.e., a region outside the obstacles). Then, they defined the problem of path planning under the condition that the CVaR of the distance does not go beyond a fixed threshold. Using this

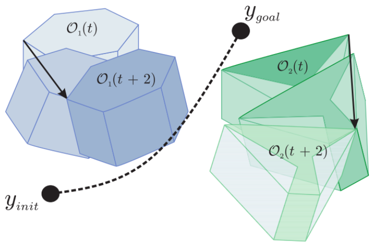


Figure 2.24: Illustration of the method, from [137]. The robot navigates between randomly moving obstacles.

method, they effectively planned trajectories for a quadrotor in a 3D environment, adjusting the safety and conservativeness of the motions.

2.4 Conclusion

In this chapter, we provided a glance of the main methods to assess risk in the main robotics applications. The risk assessment is divided into three main components, which are the traversability, collision and efficiency. The first one covers the interaction between the robot and the ground, answering whether the robot can safely navigate at this position or not. The second one focuses on the possible collisions the robot might engender while moving. Finally, the third looks at the efficiency of the task that may vary from one application to the other, from the energy consumed to the speed of the completion of the task. After a better understanding of these methods, a more in-depth discussion of the risk was presented. First, we showed that a risk metric is not an easy thing to create and that no obvious solution is available. Then, we presented some attempts to formalize it, that generalizes the aforementioned methods of Section 2.2. Whereas some frameworks succeed at proposing a risk assessment formalization, it needs much information that is not always available. Finally, a more theoretical study has shown that not all risk metrics are consistent and some are objectively better than others.

CHAPTER 3

MAPPING FRAMEWORKS

3.1 Introduction 65

3.2 Semantic Approaches 66

3.2.1 Topological Mapping 66

3.2.2 Dynamic Obstacle Detection 69

3.3 Metric Approaches 73

3.3.1 The Bayesian Occupancy Grid 75

3.4 Conclusion 90

3.1 Introduction

In the context of mobile robotics, the first key process of any autonomous system is the mapping of the environment. As the robot evolves in an unknown world, it needs to create a representation of any threat or object that might hinder its mission. For instance, a rover on Mars maps the traversability of the soil, whereas an intelligent vehicle maps the static obstacles (e.g., walls, fences) as well as dynamic obstacles such as pedestrians and cars.

In this chapter, we focus on the creation of collision-based maps that store an information of occupancy. First, we look at the semantic techniques, where each obstacle or hazard is stored as a single, known entity. This leads to the creation of topological maps, where we concentrate on the application of these maps in the context of autonomous vehicles. Once such map stores the static environment, we investigate the detection and tracking of dynamic obstacles.

Conversely, the metric maps store the information of occupancy for each given position in the environment. As such a field is not storable, the well known solution is to tessellate the environment into cells. Figure 3.1 provides example of both semantic and metric maps.

Although being more attractive in many applications, semantic maps tend to be more computationally expensive and do not work as well in case of complex, noisy environments.

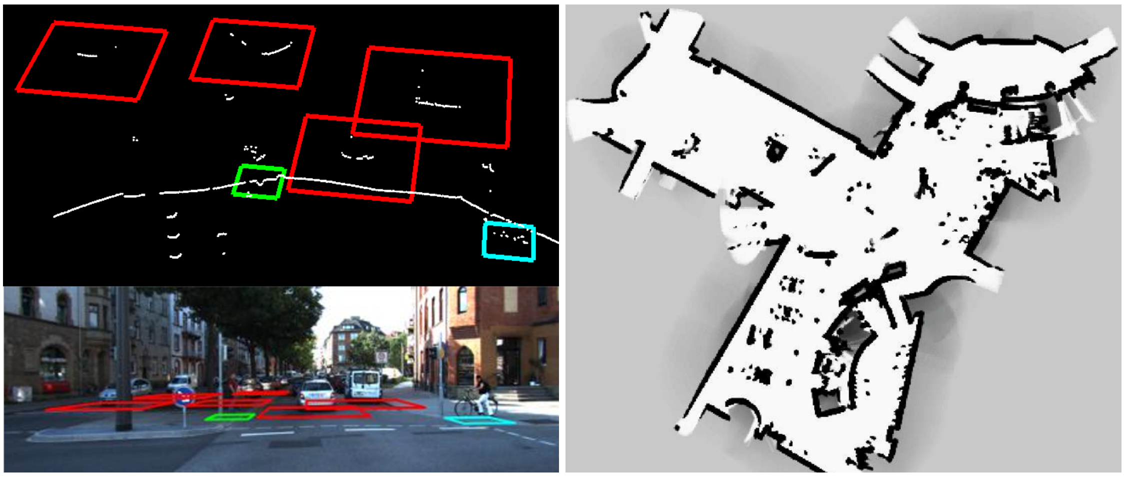


Figure 3.1: Example of semantic map (Left, from [138]) where each hazard is identified as a particular entity. On the contrary, a metric map (Right, from [1]) stores the information of occupancy for each position in the environment: the darker the cell, the higher the probability of occupancy.

3.2 Semantic Approaches

Semantic approaches choose to represent the environment as a set of entities. Entities can represent dynamic obstacles such as pedestrians or cars but can also correspond to static obstacles, such as walls or poles. In this section, we split the discussion into two distinctive parts: first, we investigate the mapping of static environment, leading to the creation of topological maps, with a focus on its application to autonomous vehicles. Then, we present a survey of the methods of detection of dynamic obstacles that can aggregate the topological map already constructed.

3.2.1 Topological Mapping

Topological maps [139]–[141] represent the environment in an abstract manner using a graph, where nodes depict distinctive elements of the scene and arc models potential relation between them. These maps have the advantage of being simple and compact, scale better and require less storage space than their counterpart, the metric maps. Garcia-Fidalgo *et al.* [142] proposed a comprehensive study of the vision-based topological mapping techniques. As suggested in their article, localization and mapping are intrinsically related.

Aynaud *et al.* [144] used topological maps to localize the robot in urban environments. Using Bayesian networks, the robot was able to infer the best feature to detect in order to precise its localization in the map. Delobel *et al.* [143] enhanced the framework by proposing to refine the map in the same time, adjusting the position of the walls. As such, the robot was capable of questioning the information contained in the map and correct them if necessary.

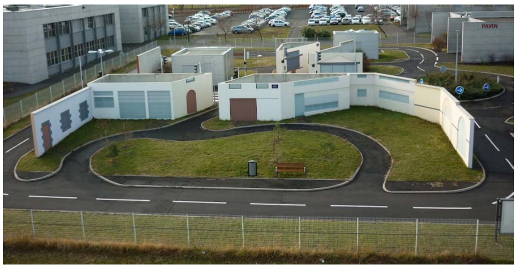


Figure 3.2: Example of urban environment. As done in [143], the elements of the environment can be stored as distinct entities that were in this case walls and roundabouts.

Representation of roads

Topological maps are fitted to represent somewhat structured environments of known features. For instance, such maps would not perform well in unstructured environments where a lot of obstacles such as bushes, tall grass and hills populate the environment. In the case of autonomous vehicles, and more particularly in the case of highway navigation, the environment is very well known and structured. The static environment simply consists of lanes where the vehicles evolve. For instance, a topological representation of this environment can be a sequence of waypoints that are the center of the lanes of the roads.

For instance, Kasmi *et al.* [146] proposed to use topological maps to localize a vehicle in its way, using digital maps such as OpenStreetMap (OSM). Because of the low definition of the aforementioned maps as well as the noise on the GPS localization, the vehicle also detects lanes to better estimate its position, as shown in Figure 3.3. Using these detections, an autonomous vehicle can easily perform localization, remapping and trajectories planning.

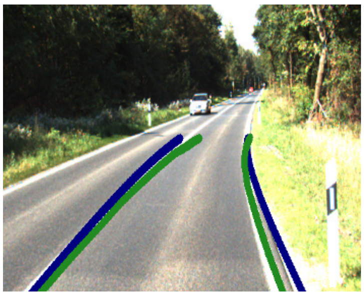


Figure 3.3: Example of lane detection for topological road mapping with a lidar (green) and a camera (blue), from [145].

In a more global point of view, such map can be created from on-board sensors, aerophotogrammetry or even satellite images. The Defense Advanced

Research Projects Agency (DARPA) challenges [147], [148] proposed more sophisticated maps containing lanes, lane crossings and lane mergers. Each lane also contained information such as its width and speed limit. Urmson *et al.* [149] used a graph model of the DARPA maps for the self-driving car Boss (Carnegie Mellon University), claiming the first place in the 2007 DARPA Urban Challenge. Each node denoted a waypoint and directional edges betoken lanes that connect the node to all other waypoints it can reach. Costs were assigned on the edges taking into account the time of traversal, length of the lane and the complexity of the environment. However, manual labelling of the road shapes were necessary for the car.

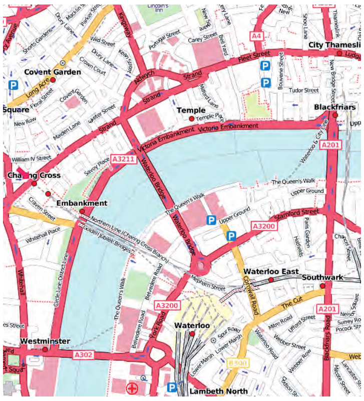


Figure 3.4: example of OSM map from [150].

In that sense, Haklay *et al.* [150] and Ramm *et al.* [151] proposed an open source, cooperative mapping framework called OpenStreetMap (OSM). OSM is a collaborative project aiming at creating a free editable map of the world. The motivation for such enterprise is the restrictions on use or the availability of map information across the world, as well as the rise of inexpensive small satellite navigation devices. The framework creates semantic maps based on three main components that are nodes, ways and relations. Other road properties are given as properties of the following elements:

Nodes are point-shaped geometric elements which are used to represent space points in terms of latitude and longitude. Moreover, Nodes are the only primitives with position information. Therefore, all other primitives depend on Nodes to locate themselves;

Ways are a list of nodes arranged in a certain order that represent a way such as roads, railways or pedestrian walkways; and

Relations are used to model logical or geographic relationship between objects (nodes or ways). For instance, a bus route is represented as a relation between several nodes representing the bus stops.

Bender *et al.* [152] proposed a high definition topological road map called lanelet map, for the self-driving vehicle Bertha. The lanelet map includes both geometrical and topological features of road networks, such as roads, lanes, and intersections, using atomic interconnected drivable road segments. For the self-driving car Bertha, the authors chose to manually annotate all the elements and properties of the map. The lanelet map was successfully tested throughout an autonomous journey of 103 km on the historic Bertha Benz Memorial Route. Because of the growing need for larger maps, Bastani *et al.* [153] proposed a method seeking to automate the generation of topological map using a Convolutional Neural Network (CNN) on aerial images. The test on aerial images covering 24 km^2 of 15 cities achieved an average error of 5% on a junction-by-junction matching metric. With the same intent, Wegner *et al.* [154] used Conditional Random Field (CRF) to model the structure of the road network, segment it into super-pixels and adding paths to connect these regions. Mnih *et al.* [155] used satellite images and CNNs to gather road segments, using the spatial coherence of the scene to improve performance.

Recently, High-Definition (HD) maps were born to tackle intelligent vehicle problematics. Such maps provide very precise (centimeter-level) maps and contain rich information about the world. However, one can easily suspect that creating HD maps is very costly. Indeed, such maps need inherent frequent and precise updates that need a lot of resources. As such, only private corporations can usually create enough data to manufacture these maps and sell them. Zoller [156] assessed and ranked the top vendors that are HERE, Google, Mapbox and TomTom.

3.2.2 Dynamic Obstacle Detection

Whereas the last section only covers the mapping of the static environment, many robotic applications also need to monitor the dynamic obstacles that populate the environment. We present in this section the main trends in dynamic obstacle detection, that can be classified into three main categories: cluster, geometric and learning based. Whereas Petrovskaya *et al.* [157] also mention a fourth category that is the grid-based methods, these frameworks are studied in Section 3.3.

Cluster-based Models

Cluster based methods rely on segmenting the environment into groups of data, called clusters, that are more similar to each other than to those in other groups. In the specific context of robotics and range measurements, the aim is to aggregate the measurements such that each group of data represents an obstacle such as a pedestrian or a car. Figure 3.5 shows an example of such clustering in the case of pedestrians crossing the road in front of the robot. One can also notice clusters on the right of the figure that correspond to a fence and a tree.

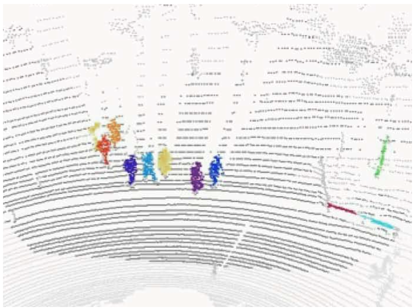


Figure 3.5: Example of clustering algorithm on lidar data, from [158].

Zhang *et al.* [159] proposed to associate to each cluster a bounding box and infer whether a cluster is a vehicle or not depending on the box size. Fayad *et al.* [160] directly took into account the shape of the detected objects to optimize the clustering step. As such, the system was able to infer the shape of the obstacles and track them over time, while fusing the estimation with a camera. Peng *et al.* [161] filtered the point cloud beforehand and perform an identification step after the clustering, leading to the labelling of three classes that are lines (e.g., a wall), circles (e.g., a pedestrian) and rectangles (e.g., a car). In a similar fashion, Catapang *et al.* [162] identified the shape of the clusters with an inexpensive 2D lidar PulsedLight LIDAR-Lite v1.

Clustering is also done using cameras: Chen *et al.* [163] used a stereo camera to cluster super-pixels in the images. Merging the super-pixels according to their boundary types and their movements, they were able to effectively detect moving obstacles in the scene. Using a fusion of a camera and a lidar, Hwang *et al.* [158] segmented obstacles in the context of intelligent vehicles, focusing on cars, pedestrians and cyclists. After each sensor independently provides obstacles detections, the results are fused and a final detection is given as a result. Finally, Matti *et al.* [164] also fused information from a lidar and a camera to detect pedestrians. First, the point cloud retrieved from the lidar was pre-processed (down-sampled and ground extraction) and obstacles were clustered. The cluster were then filtered depending on their size (e.g., a pedestrian cannot measure 3 meters). The camera classified the validated clusters and yielded a score for each detection.

Geometric-based Models

Geometric models employ non-parametric filters such as the particle filter [157]. In contrast to the last section, segmentation and association are not required steps since geometric data automatically fits data to targets. Petrovskaya *et al.* [165] presented a geometric-based method for detection and tracking of moving vehicles, as shown in Figure 3.6, that was designed for the self-driving car Junior from Stanford University [166], which finished in second place in the 2007 DARPA Urban Challenge.

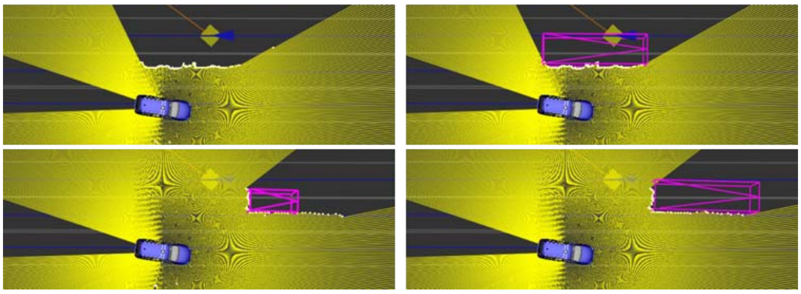


Figure 3.6: Obstacles detection (purple rectangles) from lidar measurements (in yellow) based on geometric models, from [165]. Left: Without size estimation. Right: With size estimation.

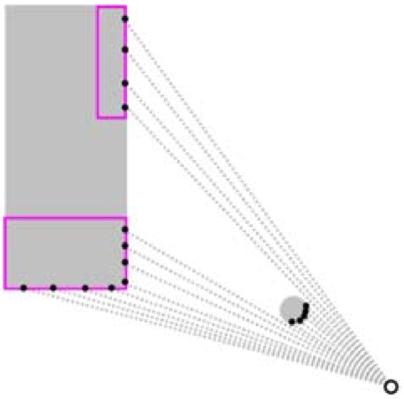


Figure 3.7: Example of scenario where a clustering algorithm would fail to estimate the true shape of the obstacles (in gray) because of an occlusion, from [165]. Geometric models do not have this problem as the shape of the obstacles is taken into account.

Using their method, the detections are more accurate than standard clustering. Indeed, as shown in Figure 3.7, clustering algorithms have troubles dealing with occlusions. In the example, the car depicted in gray is wrongfully detected as two obstacles (that can for instance be interpreted as two cyclists) and therefore can lead to hazardous decisions. Using the geometric model of a vehicle, a model-based method is able to rightfully estimate the shape of the gray vehicle. The work of Petrovskaya *et al.* [165] was improved by He *et al.* [167] to better fit the geometry and estimation the motion of the obstacles throughout the entire tracking process. The geometry of the obstacle was also better estimated by adding temporal dependencies. Ess *et al.* [168] used a stereo rig mounted on a mobile platform to categorize, track and predict the future behaviors of obstacles. Vu *et al.* [169] used a data-driven approach to interpret the laser measurements as hypothesis of moving objects trajectories over a sliding window of time. The trajectories were modelled as a sequence of detection of shapes that satisfy the motion constraints of the obstacles. Because of the high computational demands of this method, they used Monte-Carlo techniques to reach real-time capability. Wang *et al.* [170] adopted a similar approach of model-based detections but did not assume prior categories for the obstacles. Instead, a Bayes filter is responsible for joint estimation of the pose of the sensor, the geometry of the static local background, the dynamics and geometry of objects. Xu *et al.* [171] chose to model regions of interest to reduce and optimize the detection of obstacles in a context of autonomous navigation. They only look at targets inside the regions of interest and use the detections to keep a safe distance to the obstacles. Finally, Mertz *et al.* [172] proposed a detection and tracking framework relying on multiple lidar sources that can be 2D or 3D. Using a combination of segments and features in the point cloud, they associated the measurements to obstacles and keep updated a list of current obstacles.

Learning Models

Since the last decade, the learning approaches have gained tremendous popularity. Although the field started decades ago, the rise of powerful Graphical Processing Unit (GPU)s lead the field to re-birth and propose many advances in numerous applications. Many consider the field to have been reborn around 2012 with the AlexNet neural network [173] that classify images: thanks to the advance in GPU technologies, the network was able to be fully trained without any human expert in the loop, leading to a drastic increase in both computational needs and accuracy compared to state-of-the-art networks at the time. After, legion of networks have flooded the field, where Alom *et al.* [174] provided an extensive study. For instance, one can cite the You Only Look Once (YOLO) network from Plastiras *et al.* [175] and its numerous improvements over the years: not only does the network classify images, it also provides bounding boxes (i.e., position) of each detection.

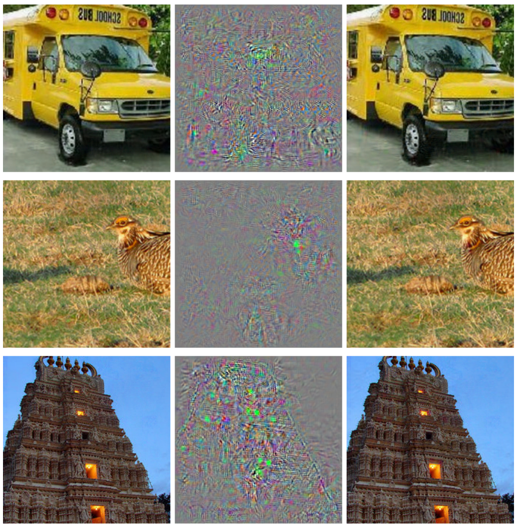


Figure 3.8: Example of incoherent results a neural network can yield in case of small perturbations, from [176]. The images (Left) have been slightly modified with an additive noise (Middle) and all yield the class ‘Ostrich’ after modification (right).

Although learning methods provide excellent results in many applications such as obstacle detection and tracking, there exist some drawbacks. The simplest and telling example is the lack of explainability and guaranties. Indeed, because of the inherent learning phase of the networks, it becomes unclear what it learnt and in which extent the network will work. In some applications, having outliers is not dangerous nor particularly bothersome, such as Optical Character Recognition (OCR). However, one can easily imagine that a network embedded in an intelligent vehicle calculating trajectories needs absolute confidence about its choices: because of a rain droplet on the camera, the network could infer incoherent choices that compromise the safety of both the occupants and the pedestrians in the scene. Figure 3.8 shows an example of what a small perturbation can do to a neural network. In this case, the network AlexNet [173] was feeding on slightly disrupted images (the perturbation is depicted in the middle column). Without the perturbation, all the images were correctly assessed by the network. However, even though a human eye cannot distinguish differences between the initial image and the disrupted one, the perturbations lead the network to infer with high confidence that all images depict an ostrich. Thus, critical systems such as autonomous vehicles need to rely on other methods to at least confirm the output of the network. Henceforth, neural networks can be seen as sensors and, as any other sensor, need to be monitored and filtered since it is prone to noise and outliers.

Théodose *et al.* [138] proposed a network built on the fusion of 3D point clouds and RGB images for 3D objects detection regardless of point cloud density. To do so, the network first takes a color image and augments the point cloud with features information. Then, the point cloud is cut into voxels and projected onto a plan to create a Bird Eye View (BEV) of the environment where detections are made. Huval *et al.* [177] evaluated deep learning algorithms for autonomous learning, concluding that the field holds promise while using the network developed by Sermanet *et al.* [178]. Mutz *et al.* [179] proposed a scenario of leader-following, demonstrating that generic learning algorithms can provide effective tracking of a vehicle without re-training for this specific scenario. The generic neural network was developed by Held *et al.* [180] and called GOTURN, for Generic Object Tracking Using Regression Networks. GOTURN is a pre-trained deep neural network capable of tracking generic objects without further training or object-specific fine-tuning.

3.3 Metric Approaches

In contrast to semantic approaches, metric maps do not require identifying entities to map the environment. Instead, they store the information of occupancy for each position in the environment. However, a field $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (i.e., an occupancy value for each nD position of the environment) has an infinite number of degrees of freedom, therefore is not storable on a machine. The most generic, popular solution is to tessellate the environment into cells of fixed size and assume that the field is constant inside the cells[1] Other methods choose to represent the field in another basis, using Gaussian processes for instance.

In this section, we provide an extensive survey of the occupancy grid method, namely the Bayesian occupancy grid framework. First, we present the main lines of the theory of the Bayesian framework that is used as a base by many works. Then, we investigate more thoroughly the different trends that emerged since the first appearance of the Bayesian Occupancy Filter (BOF) and how they augmented the framework. Figure 3.9 shows the taxonomy of the different branches of the BOF that will be explained throughout this section.

---

[1] One can also look at it from a signal processing point of view. The field is then sampled under the assumption of Nyquist–Shannon sampling theorem, meaning that the field has a maximum frequency.

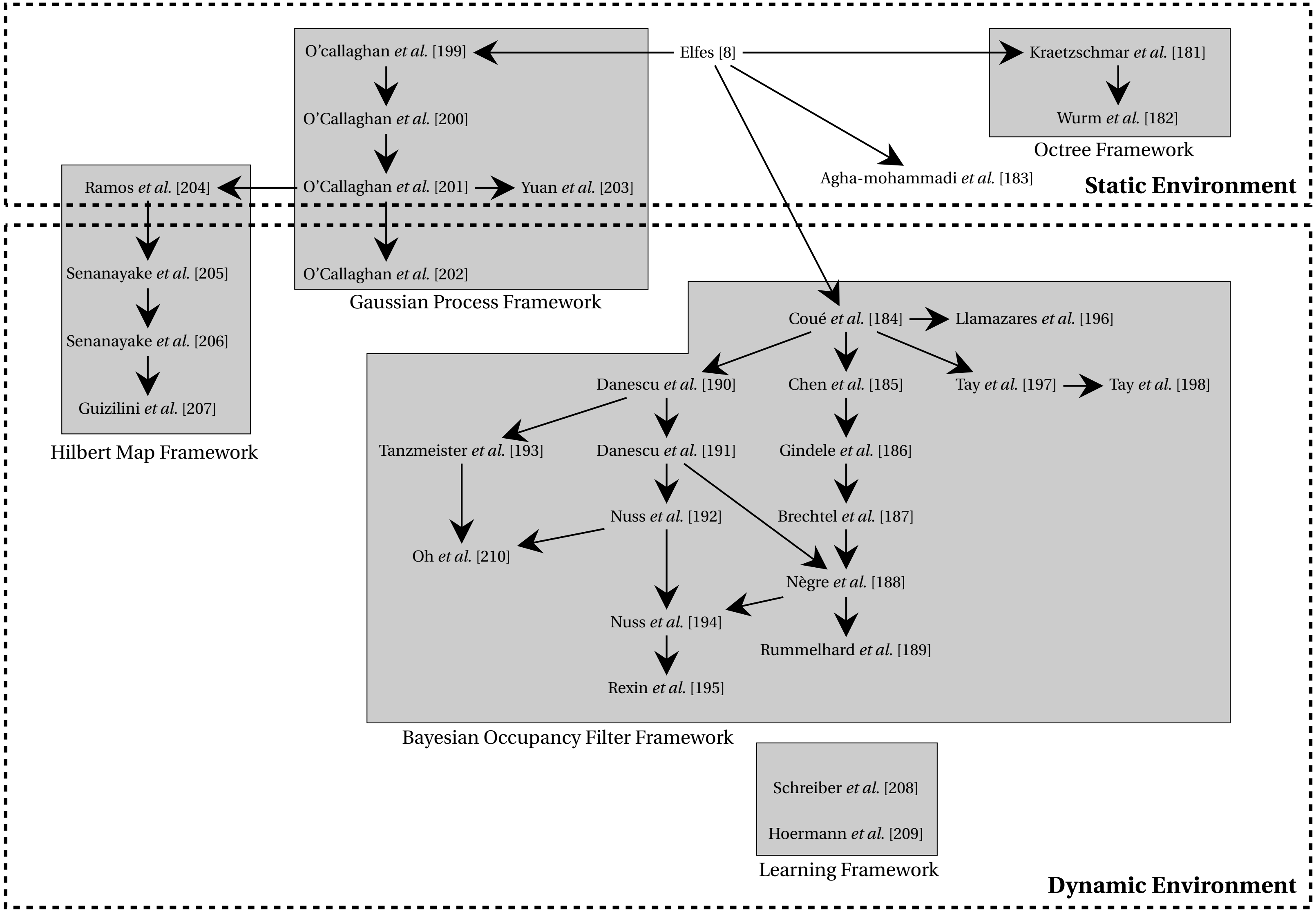

O'callaghan *et al.* [199]
Elfes [8]
Kraetzschmar *et al.* [181]
O'Callaghan *et al.* [200]
Wurm *et al.* [182]
Octree Framework
Ramos *et al.* [204]
O'Callaghan *et al.* [201]
Yuan *et al.* [203]
Agha-mohammadi *et al.* [183]
Static Environment
Senanayake *et al.* [205]
O'Callaghan *et al.* [202]
Gaussian Process Framework
Coué *et al.* [184]
Llamazares *et al.* [196]
Senanayake *et al.* [206]
Danescu *et al.* [190]
Chen *et al.* [185]
Tay *et al.* [197]
Tay *et al.* [198]
Guizilini *et al.* [207]
Hilbert Map Framework
Tanzmeister *et al.* [193]
Danescu *et al.* [191]
Gindele *et al.* [186]
Nuss *et al.* [192]
Brechtel *et al.* [187]
Oh *et al.* [210]
Nègre *et al.* [188]
Nuss *et al.* [194]
Rummelhard *et al.* [189]
Rexin *et al.* [195]
Bayesian Occupancy Filter Framework
Schreiber *et al.* [208]
Hoermann *et al.* [209]
Learning Framework
Dynamic Environment

Figure 3.9: Taxonomy of the occupancy grid framework, divided into the main trends. Each arrow represents an improvement of the framework.

3.3.1 The Bayesian Occupancy Grid

The Bayesian Occupancy Grid (BOG) has been initially proposed by Elfes [8], where many works branches out from this theory. As the static occupancy mapping while be extensively studied in the next chapters, we first explain the theory behind it as done in [1]. Then, we investigate the branches that emerged from the initial work of Elfes [8].

Estimation of the static environment

In this section, as well as for this thesis, we assume that the poses of the robot are known. That is, another framework is able to give reliable, not necessary precise, estimation of the robot's position and orientation. For instance, the Iterative Closest Point (ICP) algorithm can provide such information from lidar data. Under this assumption, the problem of mapping the environment using metric measurements is fairly straightforward. The problem can be decomposed as a graph, as shown in Figure 3.10. The robot has poses $\left\{x_t\right\}$ and measures at each step the environment, leading to the measurements $\left\{z_t\right\}$ that are dependent on the environment **m** (i.e., the map). The aim of the framework is to estimate the variable **m** given $\left\{x_t\right\}$ and $\left\{z_t\right\}$.

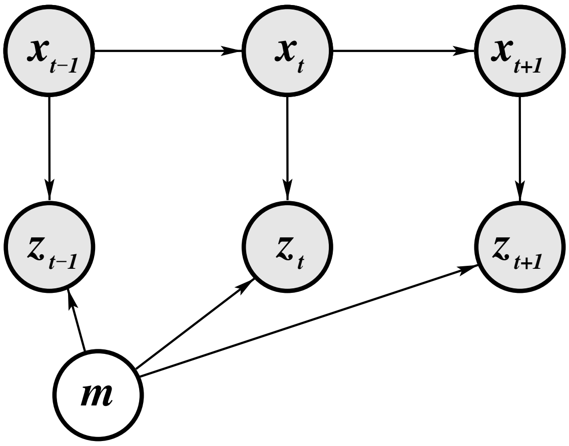


Figure 3.10: Graph modeling of the mapping problem from [1]. Each node represents a variable whereas arrows represent dependencies between them.

The environment is represented as a field **m** of random variables that correspond to the tessellation of the continuous field $f\colon \mathbb{R}^n \to \mathbb{R}$. For simplicity reasons, the following theory is presented in 2D but is easily transposable to 3D environments for Unmanned Aerial Vehicle (UAV). The map **m** is then a collection of random variables, each one representing a small portion of the environment. Mathematically speaking, we seek the posterior probability over maps given the data, that is

$$\mathbb{P}\left(\mathbf{m} \,\middle|\, z_{1:t}, x_{1:t}\right). \tag{3.1}$$

The notation $z_{1:t}, x_{1:t}$ stands for the sets $\left\{z_1, z_2, \ldots, z_t\right\}, \left\{x_1, x_2, \ldots, x_t\right\}$ that are respectively the set of the robot's measurements and the set of its poses up to the time t. As already stated, the map $\mathbf{m}$ corresponds to a field of random variables, that is

$$\mathbf{m} = \left\{m_i\right\}, \tag{3.2}$$

where $i \in [\![0, N-1]\!]$ is the index of the i-th cell, for a map tessellated into N cells. Each variable m_i can only take two values that are 'occupied' (denoted `occ`) or 'free' (denoted `free`), meaning that the cell actually contains either an obstacle or is free to cross. In order to simplify the notations, we abbreviate $\mathbb{P}\left(m_i = \texttt{occ}\right)$ by $\mathbb{P}\left(m_i\right)$ and $\mathbb{P}\left(m_i = \texttt{free}\right)$ by $\mathbb{P}\left(\neg m_i\right)$.

The problem with estimating the variable $\mathbf{m}$ is its dimensionality. Indeed, the number of cells that form the map is very large and can easily reach the tens of thousands, as the one shown in Figure 3.1. For instance, a map with 10,000 cells (e.g., a map of $10 \times 10\,\mathrm{m}^2$ with cells of size $0.1 \times 0.1\,\mathrm{m}^2$) leads to a variable $\mathbf{m}$ that can take $2^{10{,}000}$ values. As such, estimating the posterior probability of this variable is intractable. The solution is to assume that all the variables m_i are independent of each other: using this, the problem breaks down at estimating each posterior individually, that are

$$\mathbb{P}\left(m_i \,\middle|\, z_{1:t}, x_{1:t}\right), i \in [\![0, N-1]\!]. \tag{3.3}$$

Using this equation, one can use the Bayes rule to find a more convenient form:

$$\begin{aligned}
\mathbb{P}\left(m_i \,\middle|\, z_{1:t}, x_{1:t}\right) &= \frac{\mathbb{P}\left(z_t, x_t \,\middle|\, m_i, z_{1:t-1}, x_{1:t-1}\right) \mathbb{P}\left(m_i \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)}{\mathbb{P}\left(z_t, x_t \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)} \\
&= \frac{\mathbb{P}\left(z_t, x_t \,\middle|\, m_i\right) \mathbb{P}\left(m_i \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)}{\mathbb{P}\left(z_t, x_t \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)},
\end{aligned} \tag{3.4}$$

using the first Markov assumption that the state m_i is enough to estimate the robot's pose and measurement x_t, z_t. Applying a second time the Bayes rule, we obtain

$$\mathbb{P}\left(m_i \,\middle|\, z_{1:t}, x_{1:t}\right) = \frac{\mathbb{P}\left(m_i \,\middle|\, z_t, x_t\right) \mathbb{P}\left(z_t, x_t\right) \mathbb{P}\left(m_i \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)}{\mathbb{P}\left(m_i\right) \mathbb{P}\left(z_t, x_t \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)}. \tag{3.5}$$

Similarly, we have for the opposite event $\neg m_i$

$$\mathbb{P}\left(\neg m_i \,\middle|\, z_{1:t}, x_{1:t}\right) = \frac{\mathbb{P}\left(\neg m_i \,\middle|\, z_t, x_t\right) \mathbb{P}\left(z_t, x_t\right) \mathbb{P}\left(\neg m_i \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)}{\mathbb{P}\left(\neg m_i\right) \mathbb{P}\left(z_t, x_t \,\middle|\, z_{1:t-1}, x_{1:t-1}\right)}. \tag{3.6}$$

The log odds ratio of the probability yields to the cancellation of various hard to calculate terms. The log odds ratio of an event e is defined by

$$l(e) = \ln\left(\frac{\mathbb{P}(e)}{1-\mathbb{P}(e)}\right), \tag{3.7}$$

and the probability $\mathbb{P}(e)$ can be easily retrieved from the log odds ratio with the formula

$$\mathbb{P}(e) = 1 - \frac{1}{1+\exp\left(l(e)\right)}. \tag{3.8}$$

This representation wraps probabilities from $[0,1]$ to $[0,+\infty)$, yielding to an elegant update equation for the Bayes filter while avoiding truncation problems for probabilities near 0 and 1 on a computer [1].

Going back to the estimations of $\mathbb{P}\left(m_i \mid z_{1:t}, x_{1:t}\right)$ and its complement given by Equation 3.5 and Equation 3.6, we use its log odds representation $l_t\left(m_i\right)$ for the time t, leading after the various cancellations to

$$\begin{aligned} l_t\left(m_i\right) &= \ln \frac{\mathbb{P}\left(m_i \mid z_{1:t}, x_{1:t}\right)}{\mathbb{P}\left(\neg m_i \mid z_{1:t}, x_{1:t}\right)} \\ &= \underbrace{\ln \frac{\mathbb{P}\left(m_i \mid z_t, x_t\right)}{1-\mathbb{P}\left(m_i \mid z_t, x_t\right)}}_{\text{Sensor information}} - \underbrace{\ln \frac{\mathbb{P}\left(m_i\right)}{1-\mathbb{P}\left(m_i\right)}}_{\text{Prior } l_0(m_i)} + \underbrace{l_{t-1}\left(m_i\right).}_{\text{Previous estimation}} \end{aligned} \tag{3.9}$$

Oftentimes, no prior on the map is available, leading to $\mathbb{P}\left(m_i\right) = 0.5$ hence $l_0\left(m_i\right) = 0$.

The sensor information is called the inverse sensor model as it reasons from effects to causes: it provides information about the world conditioning on a measurement caused by this world. It is possible to obtain complex and accurate inverse model from the conventional measurement model, as shown in details by Thrun *et al.* [1]. Figure 3.11 shows an example of an inverse measurement model, where the darkness represents the likelihood of occupancy. Although quite simple, this type of inverse sensor model yields good results and will be used in the following of this thesis as it allows simple and fair comparison with the contributions presented later.

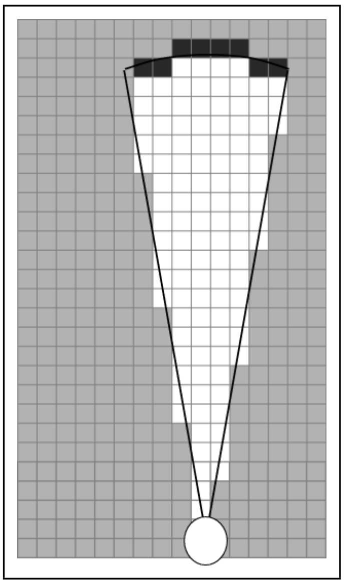


Figure 3.11: Example of measurement model from [1].

Using Equation 3.9, the robot can effectively update its belief about the environment and cre-

ate a Bayesian occupancy grid. Using this theory
as a base, many works enhanced the framework in different ways. At first, we will look at the authors who optimized the storage and processing of the map for 2D and 3D environments. Then, we investigate its applications for autonomous vehicles that started with [184] and ultimately lead to a robust framework for metric mapping in dynamic, urban environments. Finally, we present parallel works that tried to answer various limitations of the aforementioned theory such as the tessellation of the map, or tried to apply machine learning algorithm to better take into account the context.

Improvement of the spatial representation

As stated in the beginning of this section, the occupancy field $f\colon \mathbb{R}^n \to \mathbb{R}$ needs to be stored. Following this problem, the most popular solution is to tessellate the environment into cells of fixed size and assume that the field is constant inside each cell. However, representing very large environments with high resolution can still impose prohibitive memory requirements. Furthermore, representing a high frequency environment such as a garden takes as much memory than mapping an empty space, which is counter-intuitive. As such, Kraetzschmar *et al.* [181] proposed to extend the notion of Quadtrees to represent probabilistic fields more efficiently. As shown in Figure 3.12, the environment is represented as a tree where each node (i.e., a portion of the environment) is either a leaf or contains 4 nodes (i.e., sub-portions of the environment) itself: if the node contains a homogeneous value, the framework assumes that it is not necessary to subdivide this portion of the map. In the case where this area has a high variance regarding the measurements, it is then necessary to split it into smaller cells. Using this rule, Figure 3.12-Left shows the map explored in full details whereas Figure 3.12-Right shows a coarser view of it. The framework achieved to store maps with one to two orders of magnitude smaller than the respective map with fixed-sized cells.

Following this idea, Wurm *et al.* [182] proposed to represent the 3D environment using an octree, which is the 3D counterpart of the quadtree where each node if either a leaf or possesses eight children. Figure 3.13 shows an example of a 3D occupancy grid representing a tree, for different resolution levels. Each node stores the log odds ratio of occupancy and the occupancy of a node is defined in a conservative manner as the maximum log odds occupancy of its children. As such, path-planning algorithms can perform in coarser environments without additional risks. Compared to Kraetzschmar *et al.* [181] who considered several classes uniformly distributed in the probability of occupancy $[0,1]$, Wurm *et al.* [182] proposed to only consider two classes: stable and unstable nodes. A node is said to be stable if it reaches the log odds ratios thresholds $l_{\min}$ or $l_{\max}$. If a node is stable, then its children can be pruned and the node becomes a leaf. Otherwise, the node needs to be split into smaller regions and the children are kept. The results demonstrate that the approach is able to

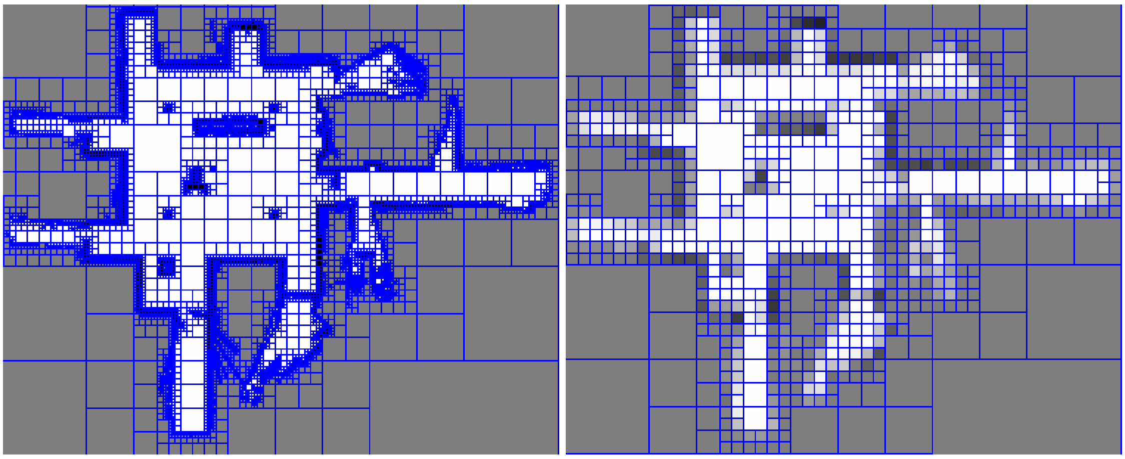


Figure 3.12: Example of probabilistic quadtrees from [181]. The quadtree on the right represents the environment with a coarser view than the one on the left.

model the environment in an accurate way and at the same time minimizes memory requirements.

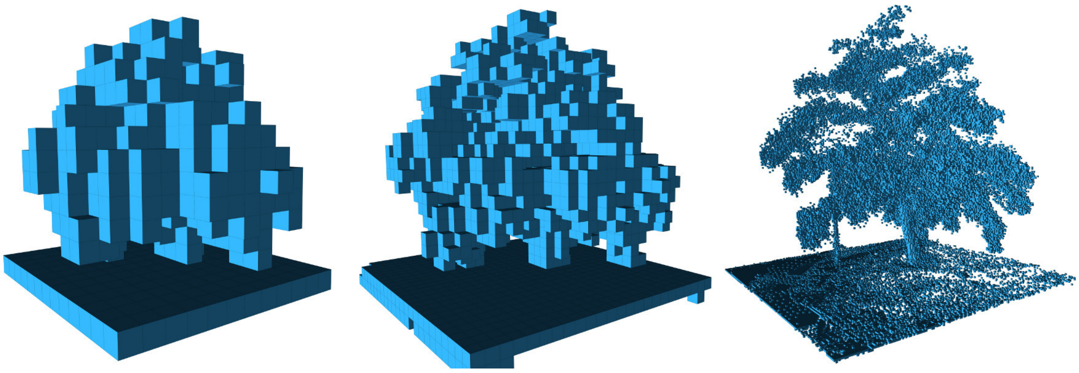


Figure 3.13: Example of octrees at different resolution representing the same environment (a tree), from [182].

The Bayesian Occupancy Filter

In this section, we look at the main trend of Figure 3.9 that developed the framework called Bayesian Occupancy Filter (BOF). Coué *et al.* [184] suggested that most obstacles tracking algorithms are based on object models. Such techniques work quite well in simple environments such as freeways, where only a few different obstacles are to be considered (e.g., cars and trucks). However, in more complex environments, these approaches usually fail as they feature numerous obstacles of very diverse shapes and behaviors. As said by Saval-Calvo *et al.* [211], multi-target tracking implies necessary knowledge of 1) whether a new sensor measurement corresponds to an existing track or not; 2) whether existing tracks should be maintained or deleted; and 3) whether new tracks should be created. This problem is intractable in urban traffic scenarios which involve numerous appearances and disappearances of numerous, rapidly evolving obstacles. Answering this problem, they proposed a metric map that combines a spatial occupancy grid with a velocity grid, leading to a four-dimensional metric space. This framework is based on Bayesian probabilities, hence the name. Nevertheless, this formulation does not reach real-time computations. Tay *et al.* [197] and Tay *et al.* [198] enhanced the initial framework of Coué *et al.* [184], where the main advantage with regard to the initial formulation is the use of velocity information to reduce the computational cost, improve predictability and reduce the overall noise in the estimation of the map.

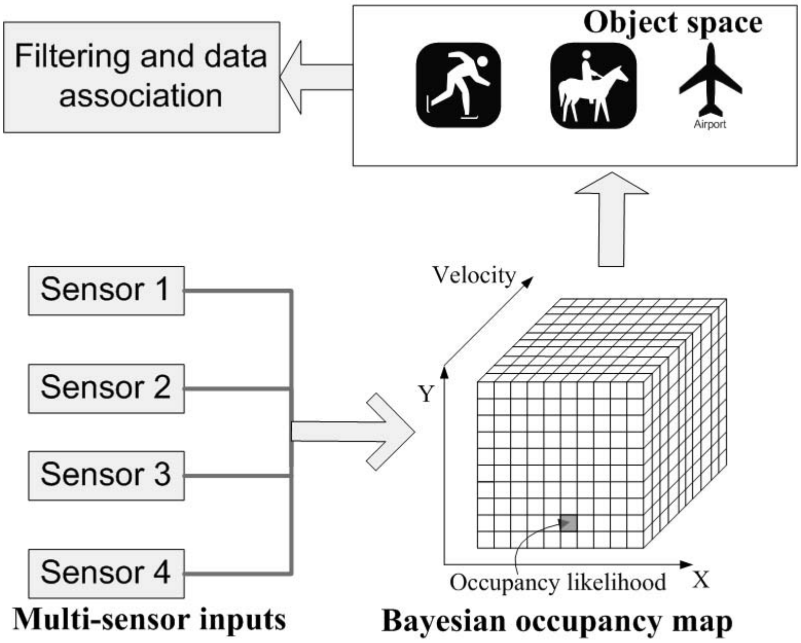
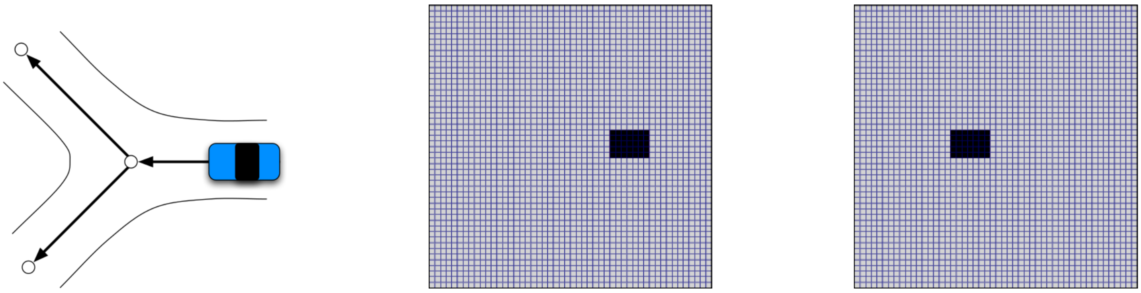


Figure 3.14: Illustration of the Bayesian Occupancy Filter (BOF), from [185]. The sensors provide measurements that are used to construct a four-dimensional grid: each 2D spatial cell has a 2D grid representing the sampled distribution of its probabilistic velocity. Chen *et al.* [185] also proposed to identify the obstacles using the Bayesian occupancy map.

Chen *et al.* [185] augmented the framework with a new formulation that significantly reduced the computational complexity and therefore enabled the framework to reach real-time capabilities. Furthermore, they show that using the BOF output to detect and extract objects from the metric space leads to a great reduction of the complexity of data association. The formulation developed by Coué *et al.* [184] and Chen *et al.* [185] leads to numerous works that branched out in various ways, as it can be seen on Figure 3.9. Llamazares *et al.* [196] extended the standard BOF that works in a two-dimensional spatial environment (i.e., a 'flat' world) by adding a third dimension that represents the height. To do so, they discretized the third dimension into three separate layers that cover the entire vertical space the robot spans in. Each layer is a 2D world that can be estimated using the standard BOF formulation.

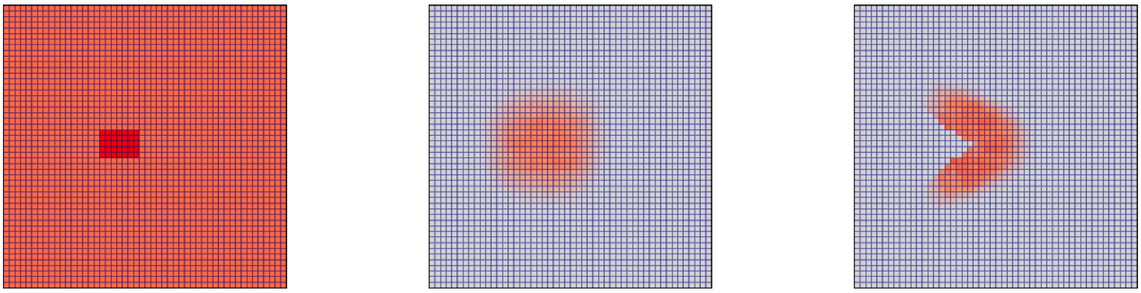
Gindele *et al.* [186] proposed a variant of the BOF using prior map knowledge, called Bayesian Occupancy Filter Using prior Map knowledge (BOFUM). Their framework predicts cell transitions more accurately by enriching the motion models of the obstacles with prior knowledge derived from the cell's context. They took advantage of the fact that object motion is often heavily dependent on its location in a scene. In the case of urban scenarios, it is more likely that a car will follow the course of a lane instead of driving perpendicular to it or on the sidewalks. These behaviour patterns can be anticipated to some degrees by looking at the geometric structure of the traffic situation. The prior knowledge of the map can be obtained from navigation systems, satellites images or simply from a road map such as OSM. Figure 3.15 shows an example of scenario where BOFUM leads to better estimations. In the case where a car is approaching a junction, a simple BOF would only estimate the future state of the car using its velocity, whereas BOFUM use the map prior to better estimate its future state, effectively splitting the occupancy into two branches at the junction.



(a) Map Information

(b) Initialization

(c) Prediction without uncertainty



(d) Simple BOF

(e) BOFUM without map knowledge

(f) BOFUM with map knowledge

Figure 3.15: Example of BOFUM framework where the information of occupancy is showed with a red scale, from [186]. (a) A car approaches a junction where it can go either left or right. (b) At this time, the occupancy grid clearly identifies the obstacle (dark square). (c) Deterministic prediction of the vehicle state. (d) Prediction using a simple BOF. (e) BOFUM prediction if no map prior is available. (f) BOFUM with map prior.

More specifically, BOFUM integrates prior knowledge in the prediction model using a reachability matrix that represents the likelihood of an object contained in a cell to reach another. In their article, they assumed three types of terrains, being {`Lane`, `Sidewalk`, `Unknown`}. Using these classes, they modeled that an obstacle which is not on the terrain `Lane` is likely a pedestrian and therefore does not have a preferred direction, whereas an obstacle in the terrain `Lane` is likely a car that will not often change lane and nor leave the road. BOFUM has been used by Brechtel *et al.* [187] where they proposed to use a similar approach to the particle filter, leading to a speedup of at-least forty times compared to the classical BOFUM. As such, the framework becomes real-time and therefore applicable in real-world scenarios. They also gave an advanced process model with motion uncertainties, showing the importance of processing uncertainties in the detected obstacles.

In parallel, Danescu *et al.* [190], [191] worked on a new approach based on particle filters to estimate the occupancies and velocities of the cells. This method will be later referred as Sequential Monte Carlo Bayesian Occupancy Filter (SMC-BOF). It is based on three main steps, that are 1) Prediction of the particles; 2) Process the measurements; and 3) Resample the particles. Nuss *et al.* [192] built on the previous framework and proposed to not only use a lidar sensor but also a Radio Detection And Ranging (radar) sensor that is able to sense velocity information using Doppler effect. Fusing the information coming from the two sensors, they showed that the map converges quicker while reducing ghost movements (i.e., stationary obstacles that are wrongly inferred as moving).

Using the work of Danescu *et al.* [190], Tanzmeister *et al.* [193] used the Dempster-Shafer theory to model the static and dynamic environment. Compared to the standard probability theory, the theory of Dempster-Shafer is able to model the unknown in the environment and can be seen as a generalization of the Bayesian theory. Oh *et al.* [210] modified the SMC-BOF to better handle conflicting information that can be caused by multi-sensors or dynamic environment, for instance. They use the concept of linear opinion pool [212] to reach a consensus over the conflictual information and yield a grid using the Dempster-Shafer theory.

Coming back to the main branch of Figure 3.9, Nègre *et al.* [188] based their work on the papers from Brechtel *et al.* [187] and Danescu *et al.* [191] and introduced the Hybrid Sampling Bayesian Occupancy Filter (HS-BOF). They presented a novel grid representation where the velocities of the dynamic obstacles are represented as particles, contrasting with the prior works [184], [185] where the velocities were discretized in a grid. Compared to the SMC-BOF[191], the method does not use particles for the static obstacles and therefore leads to a more efficient framework. In [188], the environment is represented using a mix of static and dynamic occupancy, as shown in Figure 3.16. The static environment is represented using a classical occupancy grid, whereas the dynamic obstacles are modeled with moving particles in a particle filter manner. As stated in their article, they manage to reduce the average number of particles per cell from a typical 900 to less than 2, compared to the prior works of Danescu *et al.* [191].

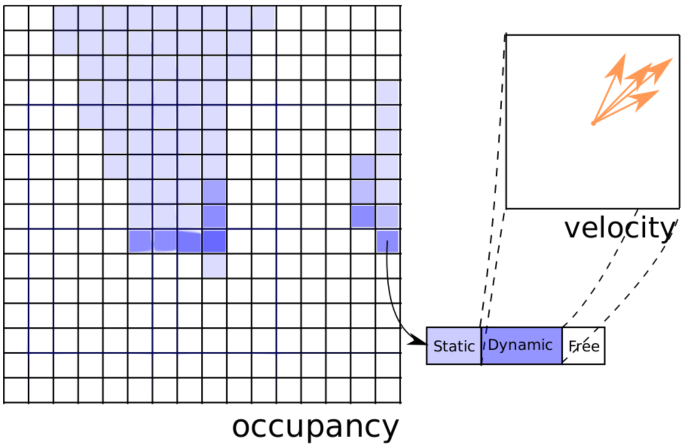


Figure 3.16: Illustration of the representation of the environment in HS-BOF, from [188]. Compared to the prior works, the velocity distribution is not discretized but approximated using particles.

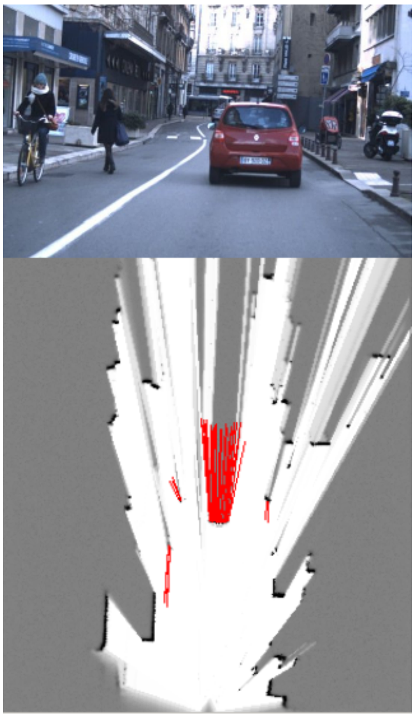


Figure 3.17: Example of dynamic occupancy map resulting from the CMCDOT framework, from [189].

Thereafter, Rummelhard *et al.* [189] based their work on the HS-BOF and created the Conditional Monte Carlo Dense Occupancy Tracker (CMCDOT), stating that the prior work of Nègre *et al.* [188] still allocates a lot of particles to irrelevant areas. They introduced a formalism to infer the true state of a cell, that is either 'occupied by a static object', 'occupied by a dynamic object', 'empty' or 'unknown'. Experimental results showed that the insertion of an 'unknown' state in the model leads to a better distribution of the dynamic samples on the observed areas and then allows to be more reactive and accurate on the velocity distributions. Furthermore, they proposed a simple clustering algorithm on the occupancy grid that allows to retrieve the different obstacles in the scene, as higher-level framework of path-planning often requires such information. Figure 3.17 depicts an example of dynamic occupancy grid created with the CMCDOT in an automotive scenario. The different obstacles in the scene, namely a car, a cyclist and two pedestrian, have their velocities accurately estimated. More specifically, they introduced a Bayesian network whose purpose is to infer in the same time the state of the cell as well as the velocity distribution in case of the cell contains a dynamic object. Figure 3.18 shows the Bayesian network: C and C^{-1} corresponds respectively to the index of the cell and its antecedent, S and S^{-1} to the state of the cell and the state of its antecedent, V and V^{-1} the velocities of the current and antecedent cells, Z the current measurement and O the occupancy of the current cell. Using this network, the expression of the problem breaks down to finding the probability $\mathbb{P}\left(SV \mid ZC\right)$, where the probability of occupancy $\mathbb{P}\left(O \mid ZC\right)$ can be trivially retrieved from this probability.

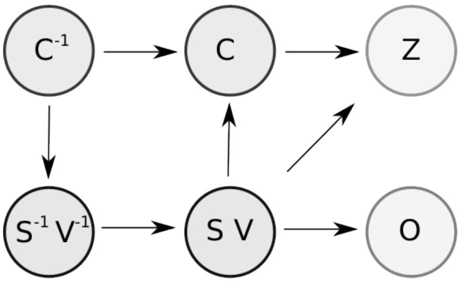


Figure 3.18: Bayesian Network used to estimate the dynamic occupancy grid in the CMCDOT approach, from [189].

In parallel, Nuss *et al.* [194] proposed another theory to create dynamic occupancy grids based on finite sets. It allows to model the environment as a stochastic, dynamic system with multiple obstacles, observed by a stochastic measurement system. They refined their previous work [192] with a better theory and fusing measurements from a lidar and a radar. In order for the framework to reach real-time capabilities, they introduced an approximation of their filter using Dempster-Shafer theory. Also, Rexin *et al.* [195] proposed Bayesian and Dempster-Shafer approaches of the work of Nuss *et al.* [194] to model occluded areas. They concluded that in overall, the Bayesian approach offers a better modeling of the free space as well as of the occluded or unknown areas, due to the presence of the particles, which are open to further potentials for the dynamic grid map. Nevertheless, the Bayesian solution requires significantly more particles than the implementation with Dempster-Shafer theory.

The Gaussian Process Framework

Up to this section, we only looked at methods that chose to discretize the occupancy field into cells. Nevertheless, it is not the only option available, and some authors ventured beyond such representations. The first successful attempt was provided by O'callaghan *et al.* [199] that represented the environment as a Gaussian Process (GP). In this section, we briefly cover the main theory of the GP, before exploring more in depth the evolution of the framework over the years.

A Gaussian Process (GP) is defined by two components: a mean function $\boldsymbol{\mu}(\mathbf{x})$ and a covariance function $\boldsymbol{\Sigma}(\mathbf{x},\mathbf{x}')$. The main idea is that for each position $\mathbf{x}$, the GP defines a Gaussian distribution at this position. Therefore, a GP can be seen as a distribution over the functions and inference directly takes place in the space of functions. In other words, we have

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x}, \mathbf{x}')\right), \tag{3.10}$$

to indicate that a continuous field is a Gaussian Process (GP). If we want to consider a variable at a specific position $\mathbf{x}^*$, we can write

$$f(\mathbf{x}^*) \sim \mathcal{N}\left(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x}, \mathbf{x}')\right), \tag{3.11}$$

The GP is used to fit a likelihood function to the training data $\left\{\mathbf{x}_i, y_i\right\}_{i=1:n}$ where $\mathbf{x}_i$ are the positions, y_i the values (in our case, the occupancy value) and n the number of training points. Without proof, it is possible to learn and query the GP that has been fitted on the training data, hence deducing a posterior mean $\hat{\boldsymbol{\mu}}(\mathbf{x})$ and covariance $\boldsymbol{\Sigma}(\mathbf{x}, \mathbf{x}')$. The reader can refer to Barfoot [213] for further theory about the GP. Figure 3.19 shows a simple example of GP in a 1D world: A robot measures the probability of occupancy of an environment composed of a free space then a wall. For each noise level on its measurements, the GP is able to infer a distribution at each position with an associated confidence interval. The higher the noise, the less confident the GP becomes and the worse the estimation is.

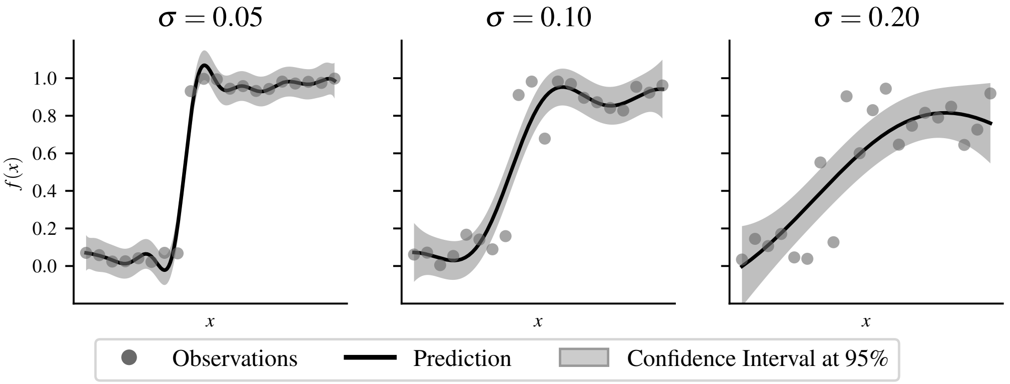


Figure 3.19: Example of Gaussian process in a 1D world for different noise level. The robot observes a wall, therefore making observation of the function $f(x)$ that corresponds to the probability of occupancy. The Gaussian Process is able to infer the mean and variance of the occupancy $f(x^*)$ for any point of interest x^* (i.e., a position).

Using this theory, O'callaghan *et al.* [199] proposed to model the occupancy of the environment in a continuous manner. Compared to the classical mapping method, this framework possesses several qualities:

1. It does not require the assumption of independencies between data points;

2. It enables accurate maps to be generated with relatively sparse sensor information;
3. It eliminates the restriction of constructing a map on a single scale; and
4. It produces an associated variance plot which could be used to highlight unexplored regions and optimize a robot's search plan.

However, the framework also suffers from the inherent issues of the GP, that is mainly the extensive computational times due to a matrix inversion, leading to a complexity of $\mathcal{O}(n^3)$ where n is the number of data points [199]. Figure 3.20 shows an example of what a GP can accomplish. Compared to the classical occupancy grid that tessellates the environment into regular cells, the GP leads to more fuzzy borders but better capture the dependency between the cells.

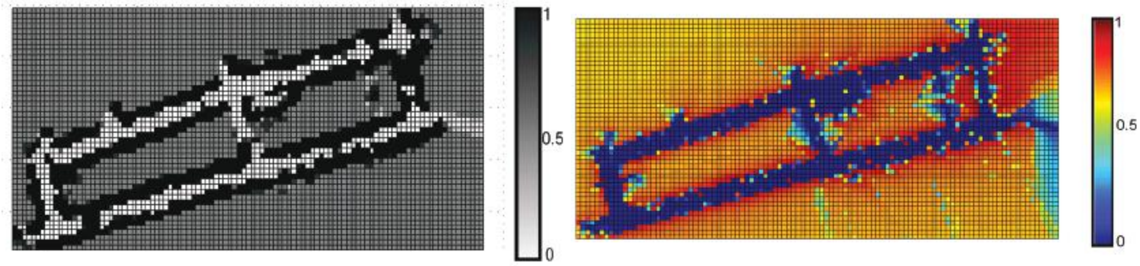


Figure 3.20: Example of large environment, from [201]. Left: mapped with a classical occupancy grid where each cell is $5 \times 5\,\mathrm{m}^2$. Right: mapped using GP.

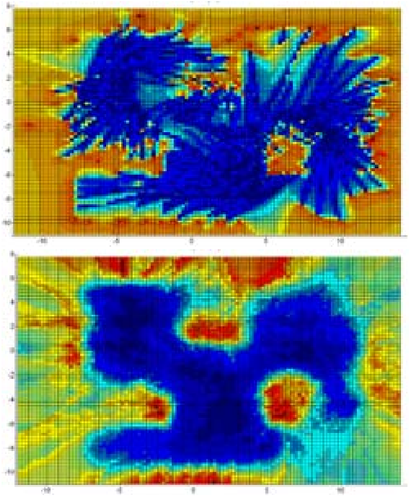


Figure 3.21: Map resulting from the initial formulation (Top) of the Gaussian Process maps [199], and the formulation from [200] (Bottom).

Following the initial work [199], O'Callaghan *et al.* [200] extended the framework to incorporate the effects of uncertainty in sensor readings and platform localisation. O'Callaghan *et al.* [201] presented a more in-depth theory that summarizes their work up to this point. Figure 3.21 shows an example of the benefits of taking into account the measurement uncertainties. As easily seen, the bottom map is less noisy and better represents the structured environment. Finally, O'Callaghan *et al.* [202] proposed to extend their framework to handle dynamic obstacles. Using techniques inspired from optical flow techniques, they developed a continuous occupancy map capable of learning static and dynamic regions and integrating observations from multiple points in time into a single continuous probabilistic spatio-temporal model of the environment. However, the problem of computation time is still present and the authors suggested the use the parallelisable properties of the framework to accelerate it. Yuan *et al.* [203] analyzed the time cost of each step of the algorithm for the static case [201]. They found that even if the training step, that has the higher complexity $\mathcal{O}(n^3)$, is considered to be the bottleneck by the community, the prediction step which is $\mathcal{O}(n^2)$ actually takes more time and is therefore the one to be optimized. They proposed to optimize the prediction step that speeds up the construction of GP maps with a factor of at least ten times.

Finally, Agha-mohammadi *et al.* [183] extended the traditional occupancy grid from Elfes [8] with a new theory that allows to partially maintain the probabilistic dependence among cells and provides confidence intervals over the occupancy information. Although retaining a tessellation of the environment, their results show that they outperform in simulation the GP framework.

Hilbert Maps

Building on the works of O'Callaghan *et al.* [201], Ramos *et al.* [204] proposed a novel method to answer the drawbacks of the GP approach. They proposed a simpler and faster approach to continuous occupancy mapping. They represented the occupancy property of the world with a linear discriminative model operating on a high-dimensional feature vector that projects observations into a reproducing kernel Hilbert space. In contrast to the Gaussian Process counterpart, the Hilbert Maps possesses an inference in $\mathcal{O}(n)$ and a learning cost of $\mathcal{O}(n)$. In Hilbert maps, the probability of occupancy for a position $\mathbf{x}$ is given by

$$\mathbb{P}\left(\mathsf{occ}\,|\,\mathbf{x},\mathbf{w}\right) = 1 - \frac{1}{1+\exp\left(\mathbf{w}^T\mathbf{x}\right)}, \tag{3.12}$$

where $\mathbf{w}$ is the parameter to learn. However, instead of learning directly in a very low-dimension space where $\mathbf{x}$ is defined, therefore a space where learning a complex environment will not be possible, they propose to project $\mathbf{x}$ into a much higher-dimensional space. As such, they apply the discriminative model not directly to the inputs $\mathbf{x}$ but to a large number of features computed from $\mathbf{x}$, denoted as $\Phi(\mathbf{x})$. Hence, the probability of occupancy is

$$\mathbb{P}\left(\mathsf{occ}\,|\,\mathbf{x},\mathbf{w}\right) = 1 - \frac{1}{1+\exp\left(\mathbf{w}^T\Phi(\mathbf{x})\right)}. \tag{3.13}$$

The key is then to understand that such mapping defines a Hilbert space of inner product $< \Phi(\mathbf{x}), \Phi(\mathbf{x}') >$. The aim is to choose $\Phi(\cdot)$ such that the inner product approximates well known kernels $k(\mathbf{x},\mathbf{x}')$ used in machine learning. In [204], they proposed several choices for the function $\Phi(\cdot)$, each of them approximating a different kernel. One can find some similarities with Support Vector Machine (SVM) techniques: the main difference is that SVMs use a different loss function in the training that leads them to be overconfident in unknown areas, as seen in Figure 3.22.

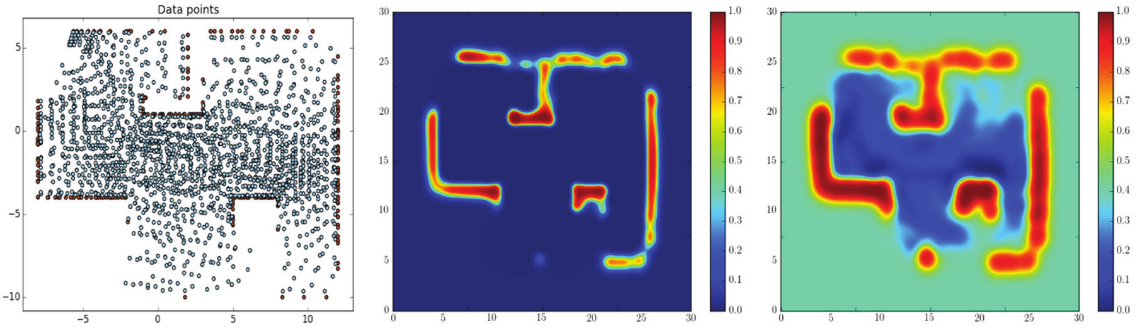


Figure 3.22: Comparison between maps produced with logistic regression and SVMs, from [204]. Left: Data points (blue is free and orange is occupied) in a structured environment. Middle: Map produced with SVMs. Right: Map produced with logistic regression from [204]. The SVM map is over confident about the occupancy status in areas with no data points, assumed unoccupied.

Senanayake *et al.* [205], [206] extended the Hilbert maps to take into account the dynamic environment. They also improved the method, removing parameters that were hard to tune. Finally, Guizilini *et al.* [207] proposed a method to capture the uncertainty of moving objects and incorporate it into the Hilbert maps. This allows the framework not only to learn the occupancy incrementally but also to predict the evolution of the dynamic environment, taking into account the uncertainties.

Learning Frameworks

In this last section, we look at a few examples of learning algorithms that work along classical dynamic occupancy grids.

Schreiber *et al.* [208] proposed to estimate a dynamic occupancy grid with a neural network that feeds on static occupancy grids. As shown in Figure 3.23, the network predicts occupancy and velocity, i.e., a dynamic occupancy grid map. The network is a combination of feedforward (red) and recurrent (blue) network layers. The Encoder-LSTM, outlined in 3.23.c), uses information of a sequence of measurement grid maps to update the internal states and estimates occupancy and velocity. The recurrent skip architecture 3.23.d) ensures dense predictions at the output. As learning approaches better cap-

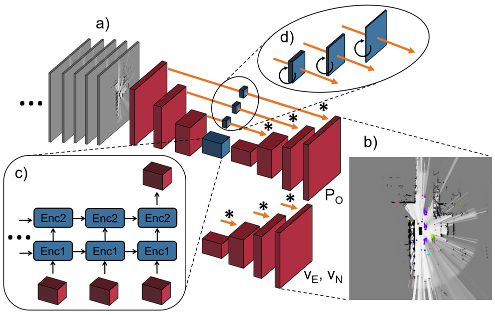


Figure 3.23: Illustration of the learning approach of Schreiber *et al.* [208], from [208].

ture the context in the scene, their approach provides more consistent velocity estimates for dynamic objects and less erroneous velocity estimates in static area.

In the same context of intelligent vehicles, Hoermann *et al.* [209] proposed a learning approach to better predict the long-term movements of dynamic obstacles. As previously said, learning approaches have the inherent characteristic of using context information, enabling the implicit modeling of road user interaction. They showed that their network is able to effectively predict movements of complex scenarios up to 3 s. The network is able to consider different maneuver classes (e.g., turn right or go straight) and interactions between road users reduce the prediction uncertainty.

Comparison of the frameworks

To conclude this section, we provide in Table 3.1 a comparison of the overall benefits and disadvantages of each type of method from Figure 3.9. Computationally speaking, the BOF framework is the fastest, with its 3D counterpart that is naturally slower due to the gain of a dimension. Nevertheless, Octomap [182] remains fast due to the usage of octrees. GP and Hilbert Maps (HM) compensate for their slower computational time by offering a continuous map. As the HM is a direct enhancement of the GP, the former is preferred when one needs continuous mapping. Their main drawback is the need of user-defined parameters that are mandatory for the learning process. Finally, learning frameworks propose an interesting approach that better take into account the context of the scene. However, as it is the case for every learning framework, an unbiased, massive dataset is needed, whereas almost no certification of performance can be provided for now.

Method	**Comp. Time**	**Continuous**	**Dimension**	**Tuning parameters**
Bayesian Occupancy Filter (BOF) [184]	++	No	2D	No
Octomap [182]	+	No	3D	No
Gaussian Process (GP) [199]	−−	Yes	2D/3D	Yes
Hilbert Maps (HM) [204]	−	Yes	2D/3D	Yes
Learning Frameworks [208], [209]	−	No	2D	Yes

Table 3.1: Comparison of the metric mapping methods in term of computational time, whether the produced map is continuous or tessellated, the dimension of the environment, and whether the framework needs user-defined tuning parameters to perform well.

3.4 Conclusion

In this chapter, we investigated the different methods of mapping. On the one hand, the semantic approach yields sparse, compressed maps that are very compelling for risk assessment as every obstacle is labelled as a distinct entity. However, such a map is not easily obtainable in complex scenarios. On the other hand, metric maps propose a more low-level approach to mapping, where the information of occupancy is stored for each position in the environment. As this field is not storable, the more popular solution is to tessellate the environment into cells of regular size, then estimate the occupancy for each of these cells. We presented an extensive list of research works that aimed at estimating the Bayesian occupancy grid and building on the work of Elfes [8]. While most of the works aimed to build the field for dynamic environments, some authors proposed in parallel other architectures that avoid the regular tessellation of the environment.

Nevertheless, the mapping of the environment is only the first link in the chain of a robotics system. In the next chapter, we look at the path-planning techniques that directly work with the aforementioned mapping algorithms.

CHAPTER 4

PATH PLANNING ALGORITHMS

4.1 Introduction **91**
4.2 Global Path Planning **93**
  4.2.1 Graph search algorithms 93
  4.2.2 Random sampling algorithms 95
  4.2.3 Interpolating Curve Planners 97
  4.2.4 Intelligent bionic algorithms 98
4.3 Local Path Planning **99**
  4.3.1 Graph search algorithms 99
  4.3.2 Sampling algorithms 102
  4.3.3 Interpolating curve algorithms 103
  4.3.4 Numerical optimization algorithms 103
4.4 Conclusion **105**

4.1 Introduction

In this chapter, we investigate the main methods of path planning, allowing robots to move in their environment. More specifically, we split the discussion into two parts that are global and local path planning.

Global path planning is the task of seeking a path from the robot to the objective that can be rather far from the robot. Such paths can be quite coarse and do not necessary take into account the robot's dynamics or mechanical constraints. For instance, in the context of intelligent vehicle, a global path planning algorithm can take an OpenStreetMap (OSM) map as input and return a path that corresponds to which roads to take to reach the target in another town.

On the contrary, local path planning algorithms compute a local trajectory that follows the path produced by the global planner. This trajectory has to take into account the dynamic and mechanical aspects of the robot, as the computed trajectory has to be acceptable by the robot (i.e., the robot is physically able to follow it). Following our example of intelligent vehicle, even if a global planner provides which road to take, a local planner is needed to compute a local trajectory at each instant. That way, the local planner is responsible for monitoring the risk of its actions.

Figure 4.1 provides a graphical example of global and local planner. Once the global planner find an acceptable path to the objective, the local planner takes over and tries to follow the path as good as possible while controlling the risk level of its actions.

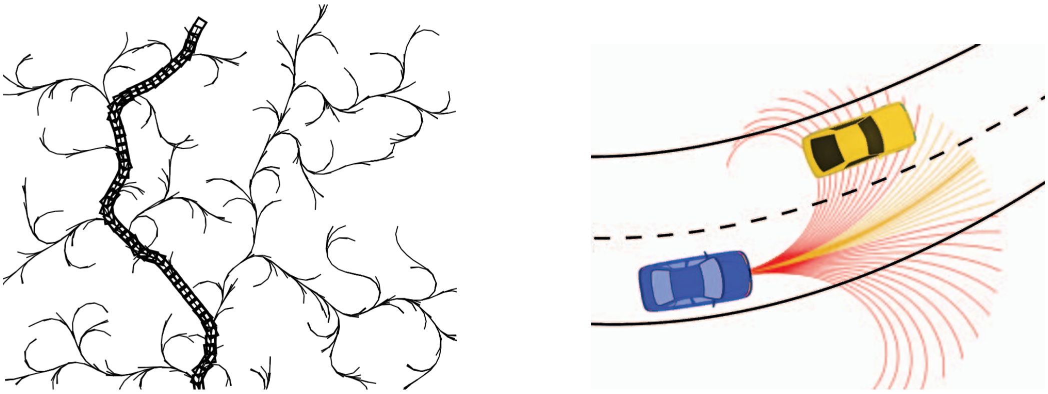


Figure 4.1: Example of global planner (Left) and local planner (Right) from respectively [214] and [215]. The task of the global planner is to provide a path to follow to the local planner, that will try to follow it as good as possible while monitoring potential dangers such as obstacles and dynamic constraints.

4.2 Global Path Planning

Global path planning algorithms have the task to find a path in complex environments. Such environments often possess local minima, meaning that convex optimization algorithm will almost always fail in this kind of setup. Per se, more complicated methods have arisen to provide more effective solutions. In this section, we investigate three main classes of global planners that are the graph search, random sampling and intelligent bionic algorithms.

4.2.1 Graph search algorithms

Graph search algorithms search the best path between the current pose of the robot and the target by converting the environment into a graph. Most of the time, the graph is simply an occupancy grid as shown in Figure 3.1. Each node corresponds to a cell and is connected to its four neighbors. Other environment representations are used, such as the underlying graph of OSM maps, or in a lesser size Voronoï diagrams [216] and probabilistic roadmaps [9]. The following figures of subsection 4.2.1 were made using the *PythonRobotics* library from Sakai *et al.* [217]. Although these algorithms are essentially used with binary maps, it is possible to use risk maps as defined in Chapter 2.

Dijkstra algorithm

Dijkstra [218] introduced the well-known graph search algorithm named Dijkstra algorithm. The algorithm finds the shortest path between a start node and a goal node of a graph. Figure 4.2 shows an example of path planning using this algorithm. In order to reach the goal, the planner expands the graph search in all directions. The cost of each edge between the node is simply the distance between them. Arnay *et al.* [219] used this algorithm to compute a path for a self-driving car. Bacha *et al.* [220] employed Dijkstra algorithm to navigate a car in and out a parking spot. In the same context, Kala *et al.* [221] proposed a multi-layer planning framework that uses Dijkstra algorithm to plan which routes and path to take. Although widely used, one can see that the algorithm expands many nodes rather far

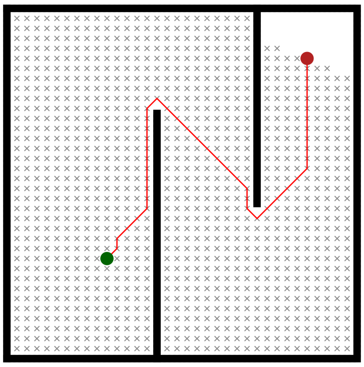


Figure 4.2: Example of Dijkstra algorithm. The explored nodes are in gray, the start and goal in green and red, and the selected path is in red.

from the goal. Using some kind of artificial intuition, a robot could choose to prioritize the nodes that are near the goal, leading to the development of the A* algorithm.

A* algorithm

The A* algorithm proposed by Hart *et al.* [222] is an extension of the Dijkstra algorithm, which speeds up the graph exploration using a heuristic function that helps search towards the goal. The heuristic has to answer certain criteria for the algorithm to provide the optimal path. Namely, the heuristic has to be admissible, meaning that for a given node, it will always underestimate the cost of reaching the goal. Although many techniques exist to produce heuristics such as problem relaxation or machine learning, the very simple heuristic of the distance as the crow flies tends to work well in practice for path planning algorithms. Indeed, for a given node, the distance for reaching the goal will always have a lower bound that is this distance, therefore the heuristic is admissible. Figure 4.3 shows an example of A* algorithm in the same environment: compared to the Dijkstra algorithm, nodes towards the objectives are prioritized and therefore the shortest path is found faster. Ziegler *et al.* [223] proposed a path planning algorithm based on A* using two heuristic functions that are 1) the rotation-translation-rotation (RTR) metric, used to model the kinematic constraints of the robot; and 2) a Voronoï based cost used to model the distance left to travel while taking into account the obstacles. Dolgov *et al.* [224] proposed the hybrid-state A* to compute a path for the robotic car Junior (Stanford University's car that finished in second place in the 2007 DARPA Urban Challenge).

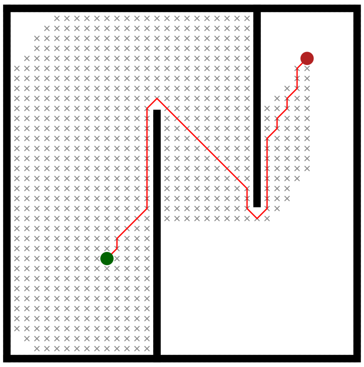


Figure 4.3: Example of A* algorithm. The explored nodes are in gray, the start and goal in green and red, and the selected path is in red.

D* algorithm

Stentz [225] proposed a variation of the A* algorithm for unknown, partially known and changing environments, called D*. The name of the algorithm, D*, was chosen because it resembles A*, except that it is dynamic in the sense that arc cost parameters can change during the problem- solving process. D* is most efficient when these changes are detected near the current starting point in the search space, which is the case with a robot equipped with an on-board sensor. Figure 4.4 shows an example of D* algorithm.

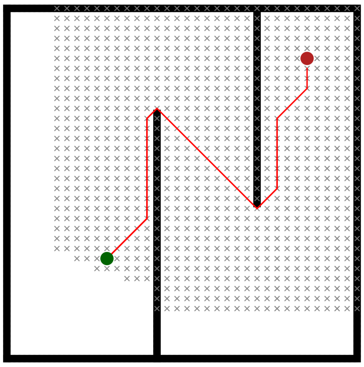


Figure 4.4: Example of D* algorithm. The start and goal in green and red, and the selected path is in red.

Urmson *et al.* [149] implemented the anytime D* algorithm to compute a path for the self-driving car Boss (Carnegie Mellon University's car that claimed first place in the 2007 DARPA Urban Challenge).

Fast Marching algorithm

As one can easily infer, the main drawback of the aforementioned methods is the necessity to represent the environment as a graph. Indeed, it leads to approximations that oftentimes overestimate the cost of a path, as a path outside the graph may be both more efficient and acceptable by the robot. In that sense, the Fast Marching method does not suffer from such issue. Coming from quantum mechanics, the eikonal equation can be used to find the shortest path in a field. Namely, the equation to solve is

$$|\nabla u(x)| = \frac{1}{f(x)}, \quad x \in \Omega \tag{4.1}$$

subject to $u|_{\partial\Omega} = 0$, where Ω is an open set in $\mathbb{R}^n$, $f\colon \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a positive real-valued function, ∇ denotes the standard gradient and $|\cdot|$ the Euclidiean norm. In the context of path planning, $f(x)$ is the acceptable speed of the robot[1] at x and the solution $u(x)$ is the shortest time possible to reach $\partial\Omega$ from x. As this differential equation is non-linear, numerical approximations have been developed and the Fast Marching method is the most popular. It discretizes the environment into a grid and solves the equation for each cell.

Compared to the Dijkstra algorithm, the Fast Marching does not assume a graph even though the solution is discretized into a grid. Thus, the calculated distance from the goal is the geodesic distance instead of a graph distance. Valero-Gomez *et al.* [226] provides an introduction to path planning using the Fast Marching algorithm.

4.2.2 Random sampling algorithms

Sampling based planning differentiates from graph based approaches as the planning occurs in the configurable space, that is the space $C \subseteq \mathbb{R}^n$ where the robot can evolve freely. As such, a prior processing of the map to distinguish between the configurable space C and its complement that corresponds to the set of unsafe pose. Sampling based planning attempts to capture the connectivity of the C-space by sampling it. This randomized approach has its advantages in terms of providing fast solutions for difficult problems. The downside is that the solutions are widely regarded as suboptimal.

---

[1] This can also represent the inverse of a cost map, where Chapter 2 provides many examples of such fields.

Barraquand *et al.* [227] introduced works that revealed to be the cornerstone of many path planning algorithms that employ randomization. Among the random samping algorithms, the most popular and used are the Probabilistic Roadmap Method (PRM) and Rapidly-exploring Random Trees (RRT) planners that we will more thoroughly explain in the next sections. For a more exhaustive survey of random sampling algorithms, the reader is invited to read the survey of Elbanhawi *et al.* [228].

Probabilistic Roadmap Method (PRM)

The method of PRM is divided into two main steps, that are the learning and query stages. At first, the space is sampled for a given amount of time. The samples that represent acceptable configurations are kept whereas the one in obstacles are discarded. Then, the query phase consists of searching a path between the start and goal configurations as a graph search in the roadmap. Roadmaps are sometimes referred to as forests, as an analogy to trees in RRT. As a result of maintaining the roadmap and specifying start and goal configurations in a subsequent stage, PRM is able to solve different instances of the problem in the same environment. It is referred to as a multi-query planner. The PRM was initially conceived for articulated robots [9], [229], [230]. Svestka *et al.* [231] extended the work for non-holonomic car-like robots. PRM are said to be probabilistically complete, meaning that if the planner was given enough time and a path exists, the framework will be able to find it.

Rapidly-exploring Random Trees (RRT)

RRT planner represents another category of sampling based planners. The aim is to grow a tree starting from the start configuration, incrementally adding new nodes until the goal is reached. At each iteration, a new node is randomly drawn from the space. If it lies in the free space, the algorithm tries to connect it to the nearest node in the tree. Just like PRM, RRT is shown to be probabilistically complete. Furthermore, RRT is faster for a single query than PRM, as there is no need of a learning phase. Nowadays, RRT algorithm has been extended in many ways such as RRT* [232]. In contrast to the standart RRT, RRT* is proven to be asymptotic optimal, meaning that it will always converge toward the optimal solution.

Fulgenzi *et al.* [110] proposed to extend the RRT to take into account probabilistic uncertainty about the environment. They defined new rules of node expansions that take into account the probability of being safe and chose the path that is the best and safe enough, i.e., for which the probability of collision is below the defined threshold.

Other Planner

For completeness, we briefly expose the other, less popular methods based on random sampling. Expansive Space Trees (EST) were developed based on the reflection that not all nodes evenly contribute to the expansion of the search tree. Unlike RRT where the sampling is uniform, EST employs a function that sets the probability of node selection based on neighboring nodes. Similar to RRT and EST, Ariadne's clew is a planner that builds a search tree to explore the search space. The difference is that it attempts to connect a node that is the furthest from the existing ones. As such, the expansion rate of the algorithm is improved compared to a standard RRT. Finally, Randomized Potential Planner (RPP) planner is used to escape local minima from potential field planner, using random walks. Table 4.1 provides a summary of the main sample based planner.

Planner	Structure	Remark
Randomized Potential Planner (RPP) [227]	Combined with Potential Field [233]	Randomly escapes local minima
Probabilistic Roadmap Method (PRM) [9], [229]–[231]	Roadmap	Sample C space, build roadmap and processes multi-query
Rapidly-exploring Random Trees (RRT) [214], [234]	Tree	Randomly samples C space and incrementally grows tree
Adiadne's Clew [235]	Tree	Connects node that is the furthest away from the other nodes
Expansive Space Trees (EST) [236]	Tree	Connects node that has more probability of expanding the search

Table 4.1: Main sample based planning algorithms, from [228].

4.2.3 Interpolating Curve Planners

In another fashion, interpolation curve planners aim at defining trajectories over a known set of points (e.g., waypoints of a route from a GPS). There exists many types of curves, in which we can cite the clothoid, the polynomial, the Bézier, and the splines [237].

Clothoid curves are defined in terms of Fresnel integrals [238]. This allows smooth transitions between straight segments to curved ones and vice-versa.

Polynomial curves [239]–[241] are commonly implemented to meet the constraints needed in the points they interpolate, i.e., they are useful in terms of fitting position, angle and curvature constraints, among others. The desired values or constraints in the beginning and ending segment will determine the coefficients of the curve.

Bézier curves are parametric curves that rely on control points. Their advantage is their low computational cost. Walton *et al.* [242] demonstrated the efficiency of Bézier curves for path planning or highway design.

Spline curves are piece-wise polynomials parametric curves divided in subintervals that can be defined as polynomial curves. This kind of curves has low computational cost, because its behavior is defined by the control points. Chu *et al.* [243] and Hu *et al.* [244] used cubic spline curves for path planning. Both of them construct a center line from a route extracted from a road network. In [244], the best path is selected considering the static safety, comfortability, and dynamic safety.

4.2.4 Intelligent bionic algorithms

Another large branch of the global path planning methods is the intelligent bionic-based method, which is a type of intelligent algorithms that simulates the evolutionary behaviors of insects. It generally includes Genetic algorithm [245], Ant Colony algorithm [246], [247], Artificial Bee Colony algorithm [248], and Particle Swarm Optimization algorithm [249]. As an example, Figure 4.5 depicts the Ant Colony algorithm: ants, represented as squares and circles, randomly walk toward the food while leaving a pheromone that dissipates over time behind them. Ants taking the shortest path therefore reach faster the goal and the pheromones of the path are stronger as they had less time to dissipate. As such, ants are more prone to take the shortest path to bring back the food to the nest as they are attracted by the pheromones. To further improve the calculation efficiency and avoid local optima problems, a lot of advanced algorithms have been proposed. Wang *et al.* [250] proposed the Optimization of the Genetic algorithm-Particle Swarm Optimization algorithm to solve the shortest collision-free path planning problem of a welding robot. Liu *et al.* [251] combined the artificial potential field and geometric local optimization method with Ant Colony algorithm to search for the globally optimal path. Mac *et al.* [252] put forward a constrained multi-objective particle swarm optimization algorithm with an acceleration methodology to generate an optimal global trajectory.

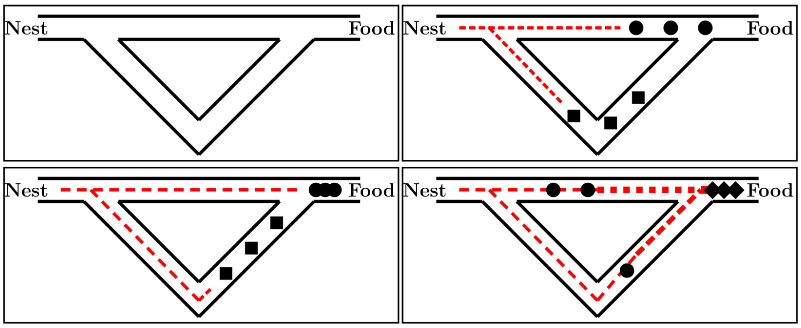


Figure 4.5: Graphical example of the ant colony algorithm, from [247]. Leaving pheromones behind them, ants taking the shortest path therefore leave a stronger trail to follow on the way back.

4.3 Local Path Planning

Once a global, coarse trajectory has been found, the robot needs to find a trajectory that locally follows the path. Depending on the global path planning algorithm used, the path can be represented as a set of points (e.g., nodes of a graph) or as a parametric curve (e.g., splines). The main goal of the local planner, sometimes called as motion planner, is to bring the robot from its current state to the current goal smoothly and safely [253]. As such, local planners need to take into account the kinematic constraints of the robot, hence provide acceptable trajectories. In this section, we present the main trends in local path planning, that are the methods based on graphs, sampling, curves and numerical optimization.

4.3.1 Graph search algorithms

Graph based algorithms for local planning are a direct extension of the global algorithms of subsection 4.2.1. In these extensions, the three most popular are the state lattice, elastic band and A* algorithms.

State Lattice

The state lattices were introduced by Pivtoraiko *et al.* [10].

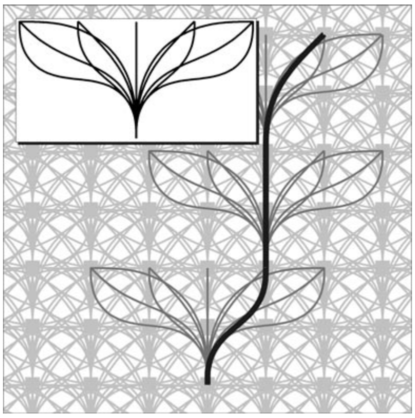


Figure 4.6: Example of state lattice with Reeds-Shepp trajectories, from [10]. Reverse motions were omitted for clarity.

A state lattice is a search graph where, opposing the global graph planner, kinematic constraints of the vehicle are taken into account. Nodes denote states of the robot whereas edge links to states that can be reached while satisfying the kinematic constraints of the robot. A feasible trajectory to the goal is therefore reduced to the shortest path problem in a graph. State lattices are able to handle dimensions such as position, velocity and acceleration. Figure 4.6 depicts an example of state lattice for Reeds-Shepp paths, meaning that the robot can either go straight, turn left at the maximum steering angle or turn right at the maximum steering angle, either forward or in reverse. The main drawback of lattice planner is the computational cost as the planner evaluates every possible solution in the graph [237].

McNaughton *et al.* [254] adapted the framework, initially developed for planetary rovers, for intelligent vehicles in highway situations. They defined a search space representation that allows the search algorithm to systematically and efficiently explore both spatial and temporal dimensions in real time. They assign to each node a state vector that contains a pose, acceleration profile, and ranges of times and velocities. The acceleration profile increases trajectory diversity at a less cost than would the finer discretization of time and velocity intervals. The ranges of times and velocities reduce computational cost by allowing the assignment of times and velocities to the graph search phase, instead of graph construction phase. Xu *et al.* [255] proposed to optimize the resulting trajectory from [254], reducing the planning time and improving the quality of the trajectory. Li *et al.* [256] built a state lattice by generating candidate paths along a route using a cubic polynomial curve. A velocity profile is also computed to be assigned to poses of the generated paths. The resulting trajectories are evaluated by a cost function and the optimal one is selected.

Elastic band

Elastic bands are proposed as the basis for a framework to close the gap between global path planning and real-time sensor-based robot control. Basically, an elastic band is a deformable collision-free path, where its initial shape is the free path generated by a planner. Subjected to artificial forces, the elastic band deforms in real time to a short and smooth path while maintaining clearance from the obstacles. The elastic continues to deform as changes in the environment are detected by sensors, enabling the robot to accommodate uncertainties and react to unexpected and moving obstacles. However, the elastic-band approach has the drawbacks of having a non-deterministic runtime and requiring a collision-free initial path.

Gu *et al.* [257] proposed a decoupled space-time trajectory planning method. The trajectory planning is divided into three phases. In the first phase, a collision-free feasible path is extracted from an elastic band, considering road and obstacles constraints. In the second phase, a velocity profile is suggested under several constraints, that are the speed limit, the obstacle proximity, the lateral acceleration and the longitudinal acceleration. Finally, given the path and the velocity profile, trajectories are computed using parametric path spirals. Trajectories are evaluated against all static and moving obstacles by simulating their future movements.

A* algorithm

Fassbender *et al.* [258] proposed to extend the classical A* algorithm for trajectory planning. They designed two novel node expansion techniques that allow nodes to be connected with feasible trajectories. Figure 4.7 shows an example of node expansion that takes into account the robot's kinematic constraints. The first one looks directly at reaching the goal from the current node, in order to help the algorithm when the standard expansion does not make good progress (e.g., in high cost regions). The second one uses a pure-pursuit controller to expand nodes along the global reference path. Following the same idea, Joachim *et al.* [259] used risk maps to plan the safest trajectory. The framework is based on the A* algorithm in combination with a kinematic model to follow non-holonomic constraints.

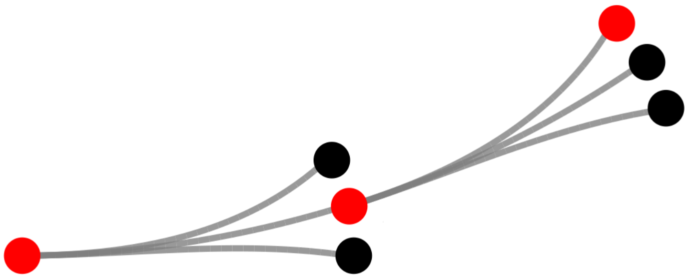


Figure 4.7: Example of node expansion from [258]. The expansion takes into account the vehicle's kinematic constraints. Expanded nodes are depicted in red.

4.3.2 Sampling algorithms

In the context of sampling based algorithms for local planning, the most used framework is Rapidly-exploring Random Trees (RRT). RRT methods for local planning [234] incrementally build a search tree from the robot's pose to the next (local) goal. The key difference is that local RRT applies an acceptable command from the nearest node to reach as close as possible the random node. As such, each directed edge represents a command applied to the robot to reach the pointed node from the source node. Figure 4.8 shows an example of RRT algorithm in the context of intelligent vehicles: a random state X_{rand} is drawn and the nearest node from the tree X_{near} is found. From this node, the algorithm creates an acceptable command that leads from X_{near} to X_{new}, the new state that is as close as possible to X_{rand}.

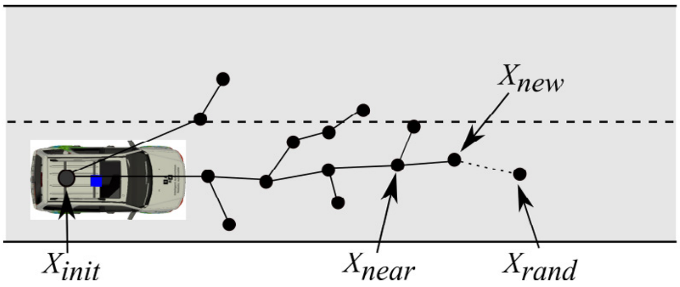


Figure 4.8: Example of local RRT, from [253]. A random state X_{rand} is drawn and the nearest node from the tree X_{near} is found. From this node, the algorithm creates an acceptable command that leads to X_{new}, the new state that is as close as possible to X_{rand}.

As easily noticeable on Figure 4.8, the main issue with local RRT planner is that the result is not curvature continuous and jerky [237]. Radaelli *et al.* [260] proposed an extension of the RRT for a self driving car. Among the modifications of the base algorithm, they biased the tree to expand toward the goal. Also, they proposed to choose the nearest node from the random state not only using a distance metric but also considering other criteria associated with environmental and traffic-law constraints, namely distance from obstacles, proximity to the lane center, maintenance of velocity limits, and execution of authorized maneuvers. As mentioned by Du *et al.* [261], RRT is often unreliable in a number of practical applications such as autonomous vehicles used for on-road driving because of the unnatural trajectory, useless sampling, and slow exploration. Following this remark, they proposed a RRT algorithm that introduces an effective guided sampling strategy based on the drivers' visual search behavior on road and a continuous-curvature smooth method based on B-spline. In the context of occupancy maps, Yang *et al.* [262] used RRT taking into account the differential constraints of the vehicle. Using this algorithm, they produced safe paths in continuous occupancy maps.

4.3.3 Interpolating curve algorithms

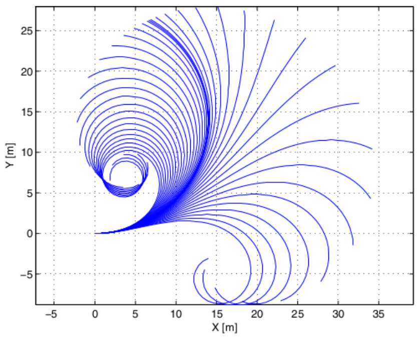


Figure 4.9: Example of clothoid tentacles for different curvatures, from [215].

Interpolating curve algorithms use a set of points and try to compute a continuous, acceptable trajectory for the robot. Also, they can take into account the comfort or dynamic constraints of the vehicle. Among the techniques, the clothoid curves, also called tentacles, are the most used. Figure 4.9 depicts some examples of such curves. Alia *et al.* [263] used clothoid tentacles for trajectory planning. Tentacles are computed for different steering angles, starting from the car's center of gravity and taking the form of clothoids. Tentacles are classified as navigable or not navigable depending on the dynamic of the vehicle and running a collision-check on an occupancy grid map. Among the navigable tentacles, the best one is chosen based on several criteria, that are the clearance from the obstacles, the change of curvature and the distance from the global trajectory. Following the same idea, Mouhagir *et al.* [215], [264] selected the best tentacle using a Markov decision process, where the reward was defined as a mixture of not colliding an obstacle and staying close of the reference trajectory. Also, they used an additional reward to skew the decision toward a left turn as overtaking on the right side is not permitted.

Himmelsbach *et al.* [265] proposed to fuse the information from a lidar and a camera to better choose the best tentacle. Indeed, inferring the traversability of a tentacle with only a lidar that produces an occupancy grid is not enough in unstructured environments such as a forest. Using visual features, the drivability of each tentacle is evaluated and the best one is chosen, leading to safer paths in off-road navigation.

4.3.4 Numerical optimization algorithms

In this last section, we present the local planner based on numerical optimization. We split the content into two parts that are the function optimization and model predictive methods.

Function optimization

Function optimization methods seek to minimize a cost function over the trajectory of the robot, considering various factors such as the collision probability or the acceleration for instance. As these methods are based on cost optimization, they can easily take into account the dynamic constraints of the robot. However, their drawback is their inherent computational cost that comes from the optimization process. Ziegler *et al.* [266] proposed a variational approach, that is finding the trajectory (i.e., function) that minimizes a cost function along the path. The cost function takes into account the distance from the driving corridor (e.g., a driving lane) and penalizes strong acceleration, jerk (i.e., acceleration changes) and high rotational velocities. Heiden *et al.* [267] used parametric trajectories defined as polynomials to navigate in occupancy grids. They define a reachability metric from the occupancy grid, then use a cost function that is a mixture of the mean and variance of the occupancy of the map. Using this cost function, they optimized the best trajectory to reach the goal.

We can also cite methods based on potential fields: obstacles are defined as repulsive forces while the goal 'attracts' the robot toward it. Vaščák [268] provides an example of path planning based of potential fields.

Model predictive methods

MPC methods try to estimate the future states of the robot and optimize along a finite time horizon. Howard *et al.* [269] proposed a method for wheeled robots that generates trajectories that takes into account numerous factors such as terrain roughness, vehicle dynamics and wheel-terrain interaction. Their framework predicts the future states of the robot and finds the best path between the initial state and the goal. They demonstrated the effectiveness of their approach for planetary rovers. Ferguson *et al.* [270] used a model predictive method for trajectory planning of self-driving vehicles. The best trajectory is selected according to their proximity to obstacles, distance to the path, smoothness, end point error, and velocity error. Li *et al.* [271] proposed a state sampling-based trajectory planning scheme that samples goal states along a route. A model predictive path planning method is applied to produce paths that connect the car's current state to the sampled goal states. A cost function that considers safety and comfort is employed to select the best trajectory. Finally, Cardoso *et al.* [272] defined the planning problem for intelligent vehicles as finding a trajectory that minimizes a cost function that takes into account the distance to the objective, the distance from the obstacles and the proximity to a given lane. Furthermore, they show that their algorithm is close to human performance, meaning that the path produced by the framework is close to what a human would do.

4.4 Conclusion

In this chapter, we presented the main trends in path planning with a focus on intelligent vehicles. Global path planning algorithm aims at finding a coarse trajectory from the current state to a far goal, whereas local planning tries to find a trajectory that locally follows the path while controlling the safety and admissibility of the decisions. Table 4.2 provides a summary of the main planning techniques used in path planning. Graph based algorithms are very popular in both global and local planning. Indeed, thanks to the discretization of the space, graph algorithms can be efficiently used at a reduced computational cost. However, most of them do not provide continuous paths and the required computational time greatly increases for those which provide it. Sampling based algorithms provide attractive results without the need of discretizing the environment. Due to their probabilistic natures, the convergence might be slow and the solution is optimal only for an infinite time. Nevertheless, they provide fast solutions that are close to the optimal one most of the time. Their biggest drawback is because of the inherent stochastic process, the trajectories are often jerky, which can be a problem for the occupants of the intelligent vehicle as well as the mechanic of the robot. Among these algorithms, Rapidly-exploring Random Trees (RRT) methods are by far the most used and many variations are available depending on the application. In local planning, tentacles are also very popular due to the easy computation and the frameworks reducing to checking which tentacle is the best according to specific criteria. Interpolating curves offer continuous trajectories at a reduced computational cost, but do not scale well in higher dimensions. In another fashion, the intelligent bionic algorithms propose interesting methods inspired from the nature. In the same way as the random sampling algorithms, they scale well in higher dimensions but suffer from high computational cost. Finally, numerical optimization methods compensate their high computational cost by allowing easy addition of constraints such as kinematic and dynamics constrains of the robot. The trajectories are also continuous, but the computational time increases greatly for each new dimension.

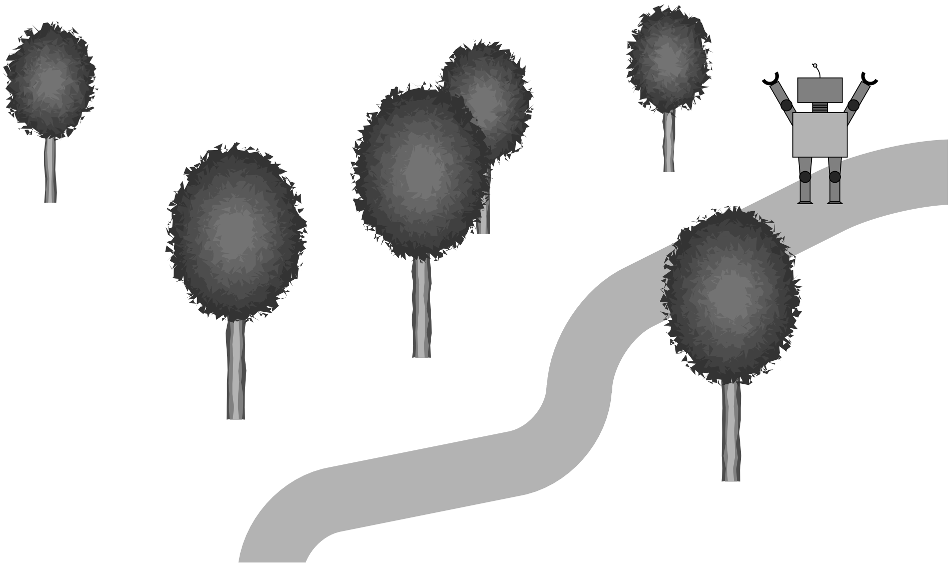
Method	Global/Local	Comp. Time	Continuous	Dimension
Graph search				
Dijkstra	●/–	–	–	–
A*/D*	●/–	+	–	–
Fast Marching	●/–	+	–	–
State Lattice	–/●	–	+	+
Elastic Band	–/●	–	+	–
Random Sampling				
PRM	●/–	–	–	+
RRT	●/●	+	~	+
Interpolating Curves				
Clothoid	●/●	–	+	–
Polynomial	●/–	+	+	–
Bézier	●/–	+	+	–
Spline	●/–	+	+	–
Intelligent Bionic				
Ant Colony	●/–	–	–	+
Numerical Optimization				
Funct. Optim.	–/●	–	+	–
MPC	–/●	–	+	–

Table 4.2: Comparison of the main planning techniques in terms of whether the method can be used for global and local planning, its computational time, whether the paths are continuous and the scalability of the method in higher dimensions. '+' means better whereas '–' means worse.

CONCLUSION

This section concludes the state of the art. In this part, we investigated the three main components of this thesis. Of course, the first and the most important is how a robot should assess a risk. Following the numerous examples and discussion of Chapter 2, we conclude that defining a framework that can unify the notion of safety is not an easy task. Even if no risk metric can surpass the other, some authors presented some tracks on what a good risk metric should represent and do. Once a risk metric is defined, the robot needs to link it to a representation of the environment. For instance, a probability of collision metric has to know every obstacle it might collide with. Therefore, we presented the two approaches in mapping that are semantic and metric maps. We concentrated on the metric fashion and more precisely on the Bayesian Occupancy Filter (BOF) for its popularity in complex, potentially dynamic environments. Finally, once a robot is capable of computing risk on this map, a way of planning and evolving in this environment is the last key component. We presented the main trends in path planning with examples of researches that incorporated risk into their planning.

In the next part, we present our attempt to generalize collision-based risk assessment in metric maps. We build upon the works that try to define a generic risk metric while proposing a novel map analogous to the BOF. Using our framework, the robot is able to assess meaningful, generic risks in metric maps without the downsides of the BOF that will be highlighted in the following.



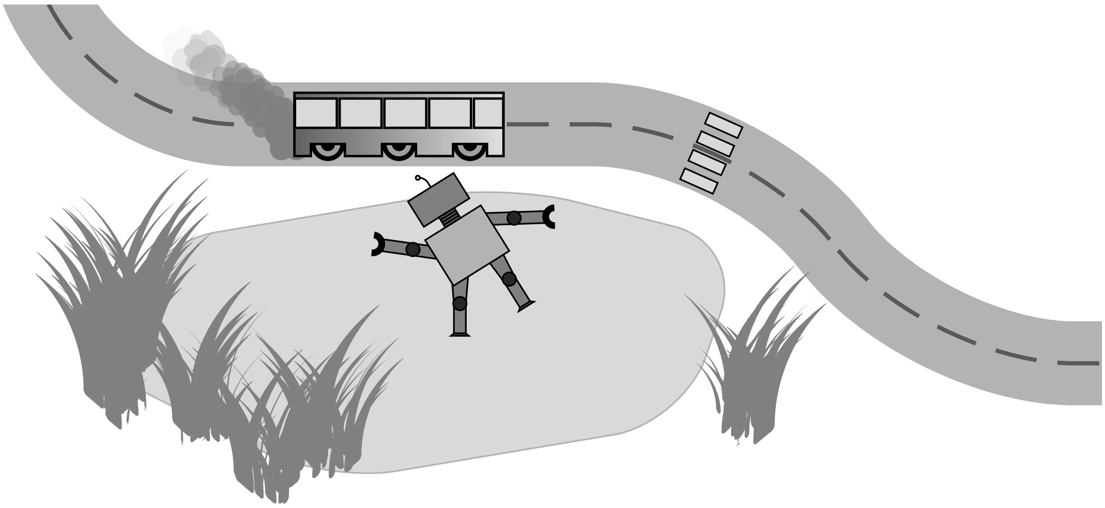
Part II

Proposed framework

MOTIVATIONS

As seen in the previous section, risk assessment for mobile robots is not an easy task. Focusing on collision-based risks, we present in this part our work that aims at theorizing risk assessment in occupancy grids. Let us consider the following example: a robot wants to cross a frozen lake with multiple grass turfs. Clearly, the robot is taking risks traversing such an environment. In the case of the frozen lake, there is a traversability-based risk that is the robot simply slips and falls, which is a very common issue in Quebec. In the case of crossing the grass turfs, the collision-based risk is that the turf hides hazardous obstacles such as boulders.

Following, the robot needs to assess the risk of each path to decide which one is the safest. As humans, we evidently prefer a path that cross the least frozen lake and the least grass turfs. Analogously, in a Bayesian occupancy grid, a robot would prefer to only cross a small portion rather than a large one of a high occupancy environment. Using this intuition, we see in the next chapter that the Bayesian occupancy grid is not able to yield consistent results because of its inherent mathematical formulation. We propose a novel framework that, in contrary of the classical Bayesian occupancy grid, does not store a probability of collision but the rate of the event. Using this formulation, complex risks can be computed and can directly be used by path planners. First, we present the framework in the static case, with an emphasize of the motivation of why such a framework is needed. The theory is validated with real-world experimentations in unstructured environments where the robot had to cross tall grass. Then, we extend the theory to the dynamic realm, where we adapt our framework to work in urban scenarios in a context of intelligent vehicles.



CHAPTER 5

STATIC LAMBDA-FIELDS FOR THE NAVIGATION IN UNSTRUCTURED ENVIRONMENTS

5.1 Context of the work 114
5.2 Theoretical background 120
5.3 Proposed approach 121
  5.3.1 Computation of the field 121
  5.3.2 Confidence intervals 123
  5.3.3 Generic framework for risk assessment 125
  5.3.4 Taking into account the mass of the obstacles 128
  5.3.5 Comparison and improvement of the reachability metric . . 132
5.4 Experimentations 134
  5.4.1 Setup 134
  5.4.2 Comparison with the Bayesian occupancy grid 136
  5.4.3 Basic path planning 145
  5.4.4 Going through tall grass 148
5.5 Discussion 151
5.6 Conclusion 153

5.1 Context of the work

In this chapter, we present our framework for risk assessment in unstructured, static environments. First, we present the overall context of this work, i.e., the problem this framework answers and how the state of the art stands to this matter. Then, we provide a theoretical section that is needed to fully apprehend the following theory. The framework is presented, followed by experiments demonstrating its utility. Finally, we conclude this chapter with a discussion of the current state of the framework.

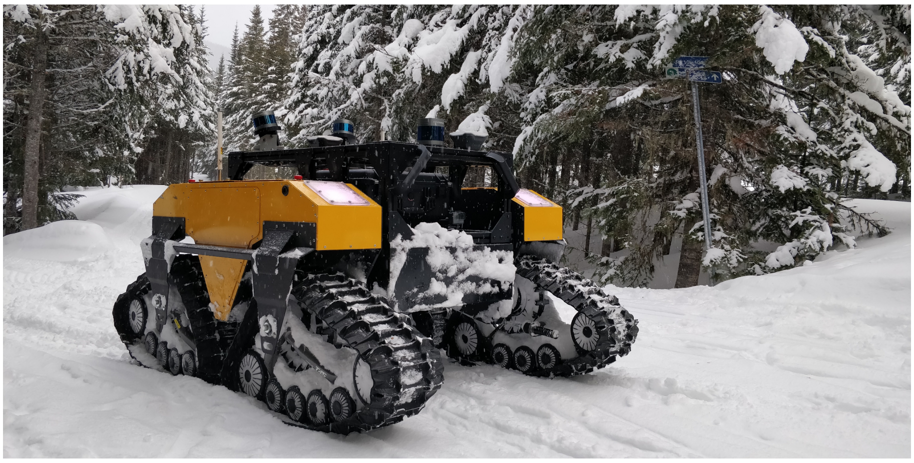


Figure 5.1: Example of unstructured environment where semantic maps would have trouble mapping all the potential obstacles. The snow banks, trees and vegetations lead to a very unstructured environments with no clear features that a semantic mapper could use.

Nowadays, autonomous robots start to gain tremendous popularity thanks to their ongoing usefulness. They start to prove themselves useful in a very broad spectrum of applications, from autonomous driving to supporting humans in dangerous jobs like mining or search & rescue missions. One common aspect of every robot's tasks is the notion of safety: before taking any action, the robots have to assess the associated risk of the action. As presented in Chapter 2, many different risks have to be considered. Whereas rovers have to watch for slippery ground, intelligent vehicles monitors constantly the possibilities of collisions. In the following, we focus on the risk of collision and leave the generalization of the framework for other types of risk in future works, as discussed in Section 5.5. In order to assess risks, the robot needs a way to represent and store the surrounding environment. In structured and controlled environments like warehouses, the easiest solution is to provide the robot with a map of the environment as well as the position of every obstacle, robot and operator. Storing such objects leads to the construction of semantic maps as described in Section 3.2, where each obstacle is stored as an object (e.g., wall, operator or robot).

Under this representation, the robot has to keep track of every moving obstacle while avoiding collisions with the environment. However, such a representation of the environment is not always available nor easy to build from raw data in all situations. For example, it is impossible to perfectly describe the underlying environment of a snowy forest. As an example, Figure 5.1 depicts a snowy forest where no clear obstacle can be extracted from raw data. Indeed, trees are covered in snow and the low vegetation, often hidden by the snow, makes any detection very noisy and possibly wrong. Clustering raw data from lidar measurements, as done by Fulgenzi *et al.* [273] for example, might not be possible for the above scenario.

Subsequently, metrics maps are used as they do not need to retrieve high level features in the environment. Instead of storing features, the metric map tessellates the environment into cells, where each one stores the information of occupancy. This kind of map has been heavily studied and used since the beginning of robotics. They were introduced by Elfes [8] who proposed the concept of occupancy grids. Section 3.3 provides an extensive review of metric mapping techniques. Among them, the Bayesian occupancy grid is the most popular and have seen numerous ameliorations over the years.

Falling back on the risk assessment problem, let us reduce for now the risk to the probability of collision. Figure 5.2 shows a Bayesian occupancy grid that the robot wants to cross and reach the goal, depicted in red. Using one of the many techniques presented in Section 4.2, the robot needs to evaluate the cost, or risk, of the paths leading to the objective. Among these paths, one is depicted in light blue. The probability of collision is therefore the probability of colliding at least one cell in the path, that is the complementary of the joint probability that all cells are free:

$$\begin{aligned}\mathbb{P}\left(\texttt{coll}_1\right) &= 1 - 0.9^6 0.4^7 0.7^5 0.9^2 \\ &\approx 0.99.\end{aligned} \tag{5.1}$$

As this equation seems at first glance reasonable, there is multiple ways to show that the above calculation is ill-formed. At first, one can imagine if the map is now stored in an octree such that the large, high occupancy bloc on the right is stored

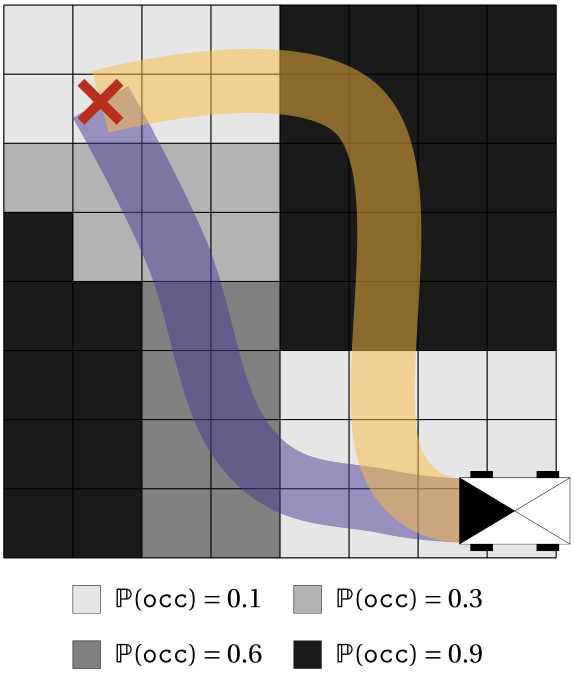


Figure 5.2: Example of occupancy grid the robot needs to cross and reach the goal in red, with two possible paths in blue and orange.

as a single cell. Under this consideration, the probability of collision of the orange path is then the probability to collide with the low occupancy cells and a single high occupancy cell, that is

$$\begin{aligned}\mathbb{P}\left(\texttt{coll}_2\right) &= 1 - 0.9^5 0.1^1 0.9^6 \\ &\approx 0.97.\end{aligned} \tag{5.2}$$

As such, given this representation of the environment, which is a totally viable one, the robot would choose the orange path that crosses a large high occupancy bloc instead of the blue one that seems intuitively better. We somewhat feel that the size of the cells have to be taken into account. To better grasp the concept, let us assume the example of Figure 5.3. The robots on the left want to cross the same environment, for instance a sparse grass field, with a constant occupancy probability of 0.1. For the first robot that tessellated the environment into four cells, the probability of collision is equal to

$$\begin{aligned}\mathbb{P}\left(\texttt{coll}_1\right) &= 1 - 0.9^4 \\ &\approx 0.34.\end{aligned} \tag{5.3}$$

In the second case, the robot did not have as much memory as the first robot and hence had to tessellate the environment in a coarser way, namely into two cells. The probability of colliding the environment for the second robot is

$$\begin{aligned}\mathbb{P}\left(\texttt{coll}_1\right) &= 1 - 0.9^4 \\ &= 0.19.\end{aligned} \tag{5.4}$$

Using these few examples, we see that the intuitive way of computing collision probability on a Bayesian occupancy grid yield counter-intuitive results.

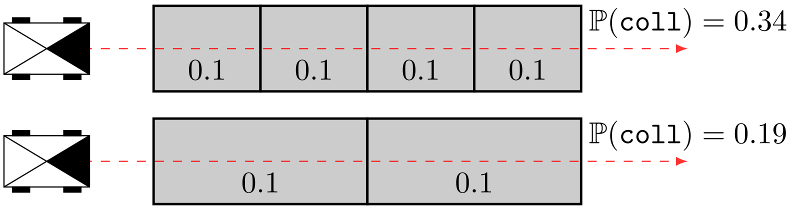

$\mathbb{P}(\texttt{coll}) = 0.34$
0.1
0.1
0.1
0.1
$\mathbb{P}(\texttt{coll}) = 0.19$
0.1
0.1

Figure 5.3: Example of collision assessment in an occupancy grid. The robots (boxes with their front represented as a filled triangle) want to cross an environment by following the dashed red line. The collision probability is uniform for the whole environment (0.1). The discretization size greatly influences the probability of collision, with the bottom scenario yielding a safer path even though the underlying environment is the same.

Several authors proposed more advanced methods to solve this issue. As shown in the last example, the problem intrinsically comes from the fact that the intuitive risk assessment method does not take into account the area of the cells. A first solution would be to "normalize" the probability of collision, as done by Lachapelle *et al.* [274]. For a given time k, assuming deterministic evolution, they model the risk as

$$R_k = \sum_c \mathbb{P}\left(c = \mathtt{occ}\right) \cdot L(\mathbf{x}_{\text{ego}}, c), \tag{5.5}$$

for the cells c the robot crosses at the time k. The function $L(\cdot)$ takes into account both the state of the robot (e.g., velocity, orientation) and the state of the cell. In their work, they define the risk as the total loss of kinetic energy of the system. However, setting $L(\cdot) = 1$ falls back to the probability of collision, and the risk becomes

$$R_k = \sum_c \mathbb{P}\left(c = \mathtt{occ}\right). \tag{5.6}$$

We see that the risk R_k is not a probability collision as it lies in $[0, +\infty)$. The risk R_k is not an estimator of a risk quantity over a probability field. Furthermore, as stated above, the risk R_k is dependant on the tessellation size: the smaller the size of the cells, the more cells the robot lies on, the more cells the sum contains and therefore the larger R_k is. Hence, they propose to multiply the risk by the area Δa of the cells:

$$R_k = \Delta a \sum_c \mathbb{P}\left(c = \mathtt{occ}\right). \tag{5.7}$$

However, this patch leads to some problems. First, one can note that the quantity R_k does not correspond to any estimator of $L(\cdot)$, as in the case of $L(\cdot) = 1$ the probability simply goes beyond 1. Second, by multiplying by the area, we lose all physical meaning of R_k. For instance, as done in their paper, they model $L(\cdot)$ as the loss of kinetic energy coming from the collision. However, due to the patch, the quantity is multiplied by the area of the cells, leading to R_k to be expressed in $\mathrm{J\,m^2}$.

Because of the non-interpretability of the risk function, it simply becomes a loss-function that the robot needs to minimize. Using this function, a path is only safer or riskier compared to another path. It is however impossible to tell whether a path is safe. For instance, we can consider a car on the edge of a cliff: in this configuration, all the paths in front of the vehicle are very hazardous. Using a loss function that is only capable of comparing paths, the robot will simply choose the path that has the minimum risk. However, in this case, even the minimum risk path is deadly. As such, the patch for this is to introduce a user-defined threshold that say whether a path is safe or not. However, the question arises how to tune such threshold as its unit is counter-intuitive.

Fan *et al.* [275] proposed to fuse risk maps for unstructured environments. They compute a risk metric as a compound of risks discretized in time and not space. The smaller the discretization, the higher is the risk. For a set of states $x_{0:N}$ and a policy π, their risk metric is defined as

$$J(x_0, \pi; m) = R_0 + \rho_0 \left(R_1 + \rho_1 \left(R_2 + \ldots + \rho_{N-1} \left(R_N\right)\right)\right), \tag{5.8}$$

with

$$\rho(R) = \mathrm{CVaR}_\alpha(R) = \inf_{z \in \mathbb{R}} \left[z + \frac{R - z}{1 - \alpha} \right] \quad \text{with } \alpha \in (0; 1), \tag{5.9}$$

where R_i depicts the risk for the time t_i. An easy intuition is to set $\alpha \to 0$ that leads the risk function $J(\cdot)$ to simplifies to

$$\begin{aligned} \lim_{\alpha \to 0} J(x_0, \pi; m) &= R_0 + \mathbb{E}\left[R_1 + \mathbb{E}\left[R_2 + \ldots + \mathbb{E}\left[R_N\right]\right]\right] \\ &= R_0 + \sum_{i=1}^{N} \mathbb{E}\left[R_i\right]. \end{aligned} \tag{5.10}$$

Thus, we easily see that the risk tends to infinity when the number of time samples N grows to infinity. As such, even if dimensional analysis tells us that $J(\cdot)$ keeps its physical unit, the computed quantity does not yield a meaningful metric. For instance, if R_i depict simply the probability of occupancy, the risk $J(\cdot)$ should lie in $[0; 1]$ and not $[0; \infty)$. This issue has repercussions: it is then impossible to have a non-homogeneous discretization (e.g., we could want to discretize less in constant, low frequency areas and more in unstructured, high frequency regions) as higher discretizations yield greater risk. Furthermore, as stated before, the cost function $J(\cdot)$ can only be used to relatively compare path and find the less risky one, even though the minimum still leads the robot into a hole.

Heiden *et al.* [267] proposed a more elegant solution to infer the probability of collision in occupancy grids. Instead of summing the probabilities, they introduce a theory based on the concept of product integrals. Product integrals can be defined as the 'product' counterpart of the classical integration. The classical Riemann integral of a function $f\colon [a, b] \to \mathbb{R}$ can be defined by the relation

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{\Delta x \to 0} \sum f(x_i) \Delta x \tag{5.11}$$

where the limit is taken over all partitions of the interval $[a, b]$ whose norms approach zero. Analogously, one can define the product integral, called Volterra type II integral, as

$$\prod_a^b f(x)^{\mathrm{d}x} = \lim_{\Delta x \to 0} \prod f(x_i)^{\Delta x} \tag{5.12}$$

It can be shown that this product integral can be computed as[1]

$$\prod_{a}^{b} f(x)^{\mathrm{d}x} = \exp\left(\int_{a}^{b} \ln f(x)\,\mathrm{d}x\right) \tag{5.13}$$

Using this theory, Heiden *et al.* [267] proposed to define the probability of collision over a path $\mathscr{P}\colon\ [0,D] \to \mathbb{R}^2$ where D is the total distance of the path, as

$$\mathbb{P}\,(\mathtt{coll}) = \prod_{0}^{L} \left[1 - p_o\left(\mathscr{P}(s)\right)\right]^{\mathrm{d}s}, \tag{5.14}$$

where $p_o\colon \mathbb{R}^2 \to [0,1]$ gives the probability of occupancy of the position $x \in \mathbb{R}^2$. As we integrate over the tessellated field, the probability of collision is not dependant on the tessellation size.

However, this theory also has several drawbacks. As it was the case for the previous work, there is no particular reason to introduce product integrals. It is here merely a tool to make the probabilities converge without being dependant on the tessellation size. Furthermore, they consider a robot reduced to a point, meaning that the robot has a wheelbase of zero.

Thus, we can see that the Bayesian occupancy grid is not made for calculating probabilities over a path and therefore is not able to assess meaningful risks, i.e., a risk that keeps its physical unit. Mathematically speaking, the issue comes from the fact that the Bayesian occupancy grid stores a probability of collision: one cannot integrate or look at a subspace and wondering what is the probability of occupancy. The Bayesian occupancy grid answers the question whether a position of the environment is occupied and not whether a subset of the environment will lead to a collision. In the following section, we provide a theoretical background about Poisson point processes, that will be shown to naturally come from the Bayesian occupancy grid when dealing with infinitesimal cell sizes. This leads to representing the environment as a stochastic process instead of a probability field that yield numerous tools for risk assessment. Then, we present our framework based on this process. The framework is able to infer probabilities of collision but also risks that do not depend on the tessellation size and also keep their physical units. Furthermore, we improve the framework of Heiden *et al.* [267] to relax the assumption of point robot and demonstrate that under this improvement, our framework is a generalization of their attempt to compute collision probabilities in Bayesian occupancy grids.

---

[1] Using Riemannian sums, see [276] for a proof.

5.2 Theoretical background

The key concept of our framework, called Lambda-Field, is its ability to assess the probability of collision inside a subset of the environment (e.g., the path of the robot), leading to the computation of a generic risk that can be adjusted depending on the scenario. To better understand the reasons for the following framework, we will first briefly demonstrate its construction. We assume that the probability of encountering a collision for a path of area Δa is $\lambda_i \Delta a$, where $\lambda_i \in \mathbb{R}_{\geq 0}$ is the 'rate' of the event collision and $\Delta a \to 0$ such that $\lambda_i \Delta a \leq 1$. The larger the intensity λ_i is, the more likely a collision will happen. In a macroscopic approach, the intensity λ_i corresponds to the expected number of collision in a cell of area $1\,\mathrm{m}^2$ and can therefore vary from 0 (i.e., the cell will never create a collision) to $+\infty$ (i.e., the cell will create an infinite amount of collisions during the traversal).

The probability of crossing N surfaces of areas Δa with a rate λ_i without collision is

$$\prod_{i=0}^{N-1} (1 - \lambda_i \Delta a). \tag{5.15}$$

Taking the limit of the path area $\Delta a \to 0$ leads to the computation of the Volterra type I product integral. For a path crossing a total area of A where each subregion of area Δa has a rate $\lambda(a)$, a being the total area crossed from the beginning, we have

$$\begin{aligned} &\lim_{\Delta a \to 0} \prod_{i=0}^{A/\Delta a} (1 - \lambda(i\Delta a)\Delta a) \\ &= \exp\left(-\int_0^A \lambda(a)\,\mathrm{d}a\right). \end{aligned} \tag{5.16}$$

A proof of Equation 5.16 can be found in [276]. The probability of encountering no collision over a path is then the probability that no event 'collision' happens in a heterogeneous Poisson point distribution of rate $\lambda(a)$. Taking the limit of a binomial distribution indeed leads to a Poisson point process distribution. Hence, the natural way of dealing with collisions in a continuous manner is to use Poisson point process distribution.

This process counts the number of events which have happened given a certain area, depending on the mathematical space. In our case, we want to count the number of the event 'collision' which could occur given a path (i.e., a subset of $\mathbb{R}^2$). We point out that the theory is here presented for 2D paths, but the extension in $\mathbb{R}^3$ is trivial, as the only change is the tessellation of the map being in 3D instead of 2D.

5.3 Proposed approach

In this section, we present the theory behind the Lambda-Fields. First, we show how to compute the field with a 2D lidar sensor. Then, we extend the framework to assess the risk of the robot's path and take into account the masses of the obstacles. Finally, a comparison with the work of Heiden *et al.* [267] is given, augmenting their theory to take into account the size of the robot and proving that under this amelioration, the Lambda-Field is a generalization of their framework.

For a positive scalar field $\lambda(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^2$, the probability to encounter at least one collision in a path $\mathscr{P} \subset \mathbb{R}^2$ is

$$\mathbb{P}\left(\texttt{coll}|\mathscr{P}\right) = 1 - \exp\left(-\int_{\mathscr{P}} \lambda(\mathbf{x})\,\mathrm{dx}\right). \tag{5.17}$$

Nonetheless, it is impossible to both compute and store the field $\lambda(\mathbf{x})$ as it has an infinite number of degrees of freedom. Hence, we discretize our field into cells in a fashion similar to Bayesian occupancy grids. Tessellating the field, the probability of collision is approximated by

$$\begin{aligned} &\mathbb{P}\left(\texttt{coll}|\mathscr{P}\right) \approx 1 - \exp\left(-\Lambda(\mathscr{C})\right) \\ &\text{with} \quad \Lambda(\mathscr{C}) = \Delta a \sum_{c_i \in \mathscr{C}} \lambda_i, \end{aligned} \tag{5.18}$$

for a path $\mathscr{P}$ crossing the cells $\mathscr{C} = \{c_0, \ldots, c_N\}$, where each cell c_i has an area of Δa and an associated lambda λ_i, which is the intensity of the cell. The lambda can be seen as a measure of the density of the cell: the higher the lambda is, the most likely a collision will happen in this cell.

Using this representation, we hereby see that the probability of collision is not dependent on the size of the cells. It is indeed the same to compute the probability of collision for crossing two cells of area $\Delta a/2$ or one cell of area Δa for a constant λ. As a concrete example, Figure 5.4 gives a path that the robot might follow, as well as the underlying cells (of area $0.04\,\mathrm{m}^2$) it crosses. The robot crosses 58 cells with $\lambda_i = 0.1$ and one cell with $\lambda_i = 2$. Using Equation 5.18, the probability of collision is evaluated at 0.27. One can note that this probability of collision is independent of the tesselation size while naturally arising from the theory.

5.3.1 Computation of the field

As we established a new approach to represent the occupancy of an environment, we need to develop a way to dynamically compute the lambdas. We assume that the

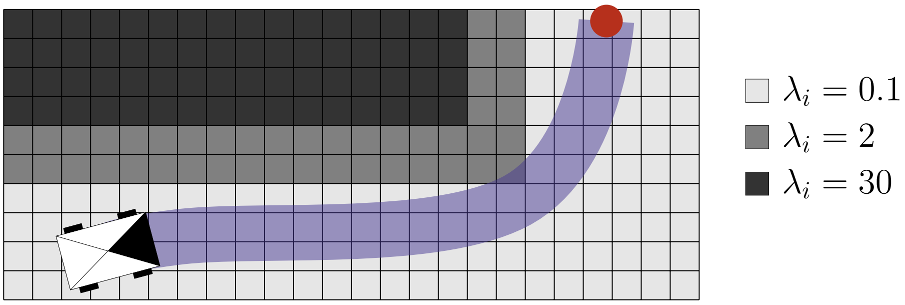


Figure 5.4: The robot wants to go to the position in red. In blue, the actual path the robot follows. Each cell has an area of $\Delta a = 0.04\,\mathrm{m}^2$. Using Equation 5.18, the probability of collision in this path is 0.27.

robot is equipped with a lidar sensor, which gives a list of cells crossed by beams without collision, and another list of cells where the beams collided. Using this sensor model, we construct the Lambda-Field in the following manner. We want to find the combination of $\lambda = \{\lambda_i\}_{i\in[\![0,C-1]\!]}$, for a map tessellated into C cells, that maximizes the expectation of the K beams the lidar has shot since the beginning. Also, each lidar beam has an associated error region $\mathscr{E}_k$ of area e_k centered on the measurement, meaning that the actual obstacle is in $\mathscr{E}_k$. Figure 5.5 shows an example of such lidar beam error region.

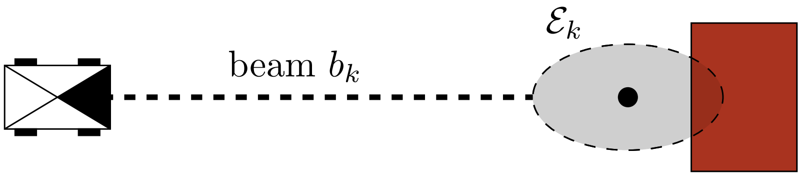


Figure 5.5: The robot measures the distance to an obstacle using a lidar sensor. The obstacle (in red) is in the area $\mathscr{E}_k$ (in gray) centered on the measurement (black dot).

Therefore, each lidar collision gives a region where an obstacle is. This kind of sensor simplification is common and is used for example by [277]. At this stage, we assumed that every lidar measurement possesses the same error region area e. The case where each beam has a different error region is dealt with in Appendix A, which can be useful for radar measurements or lidars with substantial beam divergence. For each lidar beam b_k, the beam crossed without collision the cells $c_m \in \mathscr{M}_k$ and hit an obstacle contained in the cells $c_h \in \mathscr{E}_k$. The log-likelihood of the beam b_k is

$$\begin{aligned}\mathscr{L}(b_i|\lambda) &= \ln\left[\exp\left(-\Delta a \sum_{m\in\mathscr{C}_M} \lambda_m\right)\cdot\left(1-\exp\left(-\Delta a \sum_{h\in\mathscr{E}_i} \lambda_h\right)\right)\right] \\ &= \ln\left[\exp\left(-\Lambda(\mathscr{M}_k)\right)\left(1-\exp\left(-\Lambda(\mathscr{E}_k)\right)\right)\right].\end{aligned} \tag{5.19}$$

The log-likelihood of K lidar beams is then

$$\begin{aligned}\mathscr{L}(\{b_k\}_{0:K-1}|\lambda) &= \sum_{k=0}^{K-1} \mathscr{L}(b_k|\lambda)\\ &= \sum_{k=0}^{K-1} \left[-\Lambda(\mathscr{M}_k) + \ln\left(1-\exp\left(-\Lambda(\mathscr{E}_k)\right)\right)\right].\end{aligned} \tag{5.20}$$

We want to maximize this quantity, hence nullify its derivative as the function is concave. In order to find a closed-form, we approximate the derivative with the assumption that the variation of lambda inside the error region of the lidar is small enough to be negligible. Thus, for each $\lambda_i \in \mathscr{E}_k$ we have

$$\Delta a \sum_{c_h \in \mathscr{E}_k} \lambda_h \approx e\lambda_i. \tag{5.21}$$

Using this approximation, the derivative is

$$\frac{\partial \mathscr{L}(\{b_k\}_{0:K-1}|\lambda)}{\partial \lambda_i} \approx -m_i \cdot \Delta a + h_i \frac{\Delta a}{\exp(e\lambda_i)-1}, \tag{5.22}$$

where m_i is the number of times the cell c_i has been counted as 'miss' (i.e., was outside the error region) and h_i is the number of times the cell c_i has been counted as 'hit' (i.e., was in the error region of the sensor). We finally find the zero of the derivative, leading to

$$\lambda_i = \frac{1}{e}\ln\left(1+\frac{h_i}{m_i}\right). \tag{5.23}$$

This closed-form allows a low computation complexity of the Lambda-Field. We also see that the formula is independent of the size of the cells, which is the main limitation of current representation we were aiming at resolving.

We are then able to construct the Lambda-Field using Equation 5.23.

5.3.2 Confidence intervals

In the same way as [183], we define the notion of confidence over the values in the Lambda-Field. Indeed, the more the cells are measured, the greater the confidence in the robot's movements should be. For each cell c_i, we seek the bounds λ_L and λ_U such that

$$\begin{aligned}&\mathbb{P}(\lambda_L \leq \lambda_i \leq \lambda_U) \geq 95\,\%\\ \Leftrightarrow &\mathbb{P}(\lambda_L \leq \frac{1}{e}\ln\left(1+\frac{h_i}{m_i}\right) \leq \lambda_U) \geq 95\,\%.\end{aligned} \tag{5.24}$$

To compute those bounds, we introduce the notion of false positives and false negatives: every cell measurement j has a probability p_j^h to rightfully read ‘hit’ and a probability p_j^m to rightfully read ‘miss’. The probabilities p_j^h and p_j^m have to be experimentally computed and can vary according to a great number of parameters: for example, the probability p_j^h is lower in the event of heavy rain or snow.

Using the relation $h_i = M - m_i$ where M is the number of times the cell has been measured, we can rewrite the above equation as

$$\mathbb{P}(K_L \leq h_i \leq K_U) \geq 95\,\%, \tag{5.25}$$

such that

$$\begin{aligned} \lambda_L &= \frac{1}{e}\ln\left(\frac{K_L}{M-K_L}+1\right), \\ \lambda_U &= \frac{1}{e}\ln\left(\frac{K_U}{M-K_U}+1\right). \end{aligned} \tag{5.26}$$

The quantity h_i can be seen as a sum of M Bernoulli distributions, such that

$$h_i = \sum_{j=0}^{h_i-1} \bar{h}_j + \sum_{j=0}^{m_i-1} \left(1-\bar{m}_j\right), \tag{5.27}$$

where $\bar{h}_j$ and $\bar{m}_j$ are Bernoulli variables equal to 1 if the reading was right and 0 otherwise. The quantity $\sum_j \left(1-\bar{m}_j\right)$ is hence the number of times the sensor wrongfully reads ‘hit’ instead of ‘miss’.

The distribution of h_i is not binomial, but a Poisson binomial distribution with poor behaviors in terms of computation. Since the Poisson binomial distribution satisfies the Lyapunov central limit theorem, we can approximate its distribution with a Gaussian distribution of same mean and variance:

$$\begin{aligned} \mu &= \sum_{j=0}^{h_i-1} p_j^h + \sum_{j=0}^{m_i-1} 1-p_j^m \quad \text{and} \\ \sigma^2 &= \sum_{j=0}^{h_i-1} p_j^h(1-p_j^h) + \sum_{j=0}^{m_i-1} p_j^m(1-p_j^m). \end{aligned} \tag{5.28}$$

We can then have the bounds at 95 % for example, with

$$\begin{aligned} K_L &\approx \max(\mu - 1.96\sigma, 0), \\ K_U &\approx \min(\mu + 1.96\sigma, M). \end{aligned} \tag{5.29}$$

The bounds λ_L and λ_U are then retrieved from K_L and K_U using Equation 5.26.

As an example of this bound computation, Figure 5.6 shows the behavior of the confidence interval for different confidences, where the probability of a wrong measurement is the same for every measurement. The lidar measures an empty cell c_i. The confidence interval quickly decreases as the number of 'miss' readings increases. At the 40th measurement, the lidar misreads and returns a 'hit' for the cell. The confidence interval grows around the expected lambda computed with Equation 5.23 before re-converging.

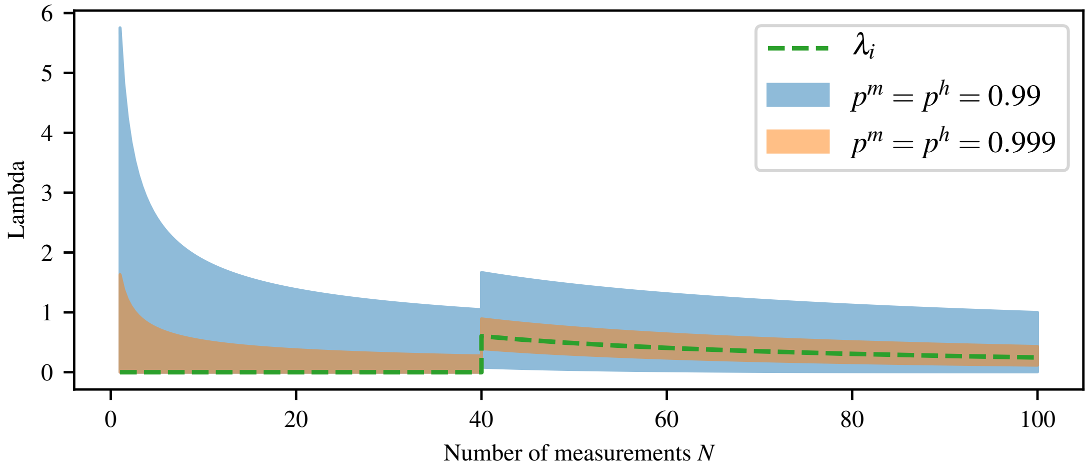


Figure 5.6: Convergence of the confidence intervals for a free cell c_i. At the fortieth measurement, the sensor misreads the cell and returns a 'hit'. The confidence interval grows around the expected λ_i before re-converging.

5.3.3 Generic framework for risk assessment

As mentioned before, the motivation of the Lambda-Fields is its capability to compute path integrals, hence a risk along a path. This risk can be defined depending on the application and is independent of the following framework, meaning that it can be interchanged without any modification of the theory. For a path $\mathscr{P} \subset \mathbb{R}^2$ crossing the cells $\mathscr{C} = \{c_i\}_{0:N}$ in order, the probability density function (p.d.f) over the Lambda-Field is

$$f(a) = \exp\left(n\Delta a\lambda_n - \Delta a \sum_{i=0}^{n-1} \lambda_i\right) \cdot \lambda_n \exp\left(-a\lambda_n\right), \tag{5.30}$$

where $n = \lfloor a/\Delta a \rfloor$ and $\lfloor \cdot \rfloor$ is the standard floor function. The variable a denotes the area the robot has crossed. Figure 5.7 shows an example of the probability density for a given path on a Lambda-Field: when the robot goes through high-lambda cells, the cumulative distribution probability quickly increases to one. One can note that it is quite easy to convert a into the curvilinear abscissa, which is far more convenient to link to the speed. For a robot of width W which has crossed an area a, its curvilinear abscissa s equals to

$$s = \frac{a}{W}. \tag{5.31}$$

Furthermore, Equation 5.30 can be easily proved as integrating $f(a)$ over a certain path $\mathscr{P}$ crossing the cells $\mathscr{C} = \{c_i\}_{0:N-1}$ gives the probability of encountering at least one collision:

$$\mathbb{P}\left(\texttt{coll}|\mathscr{P}\right) = \int_0^{N\Delta a} f(a)\,\mathrm{d}a = 1 - \exp\left(-\Lambda(\mathscr{C})\right). \tag{5.32}$$

We can then define the expectation of a risk function $r(\cdot)$ over the path, defined as

$$\mathbb{E}[r(A)] = \int_0^{N\Delta a} f(a)r(a)\,\mathrm{d}a. \tag{5.33}$$

The random variable A denotes the crossed area at which the first event 'collision' occurs. If the cells are small, we can assume that the function $r(\cdot)$ is constant inside each cell. Using this assumption, we simplify the above equation to

$$\begin{aligned} &\mathbb{E}[r(A)] = \sum_{i=0}^{N-1} K_i r(\Delta a i), \\ &\quad \text{with } K_i = \exp\left(-\Lambda(\{c_j\}_{0:i-1})\right)\left[1 - \exp\left(-\Lambda(\{c_i\})\right)\right], \end{aligned} \tag{5.34}$$

for a path $\mathscr{P}$ going through the cells $\{c_i\}_{0:N-1}$. Note that $\{c_i\}$ is a singleton whereas $\{c_j\}_{0:i-1}$ contains i elements.

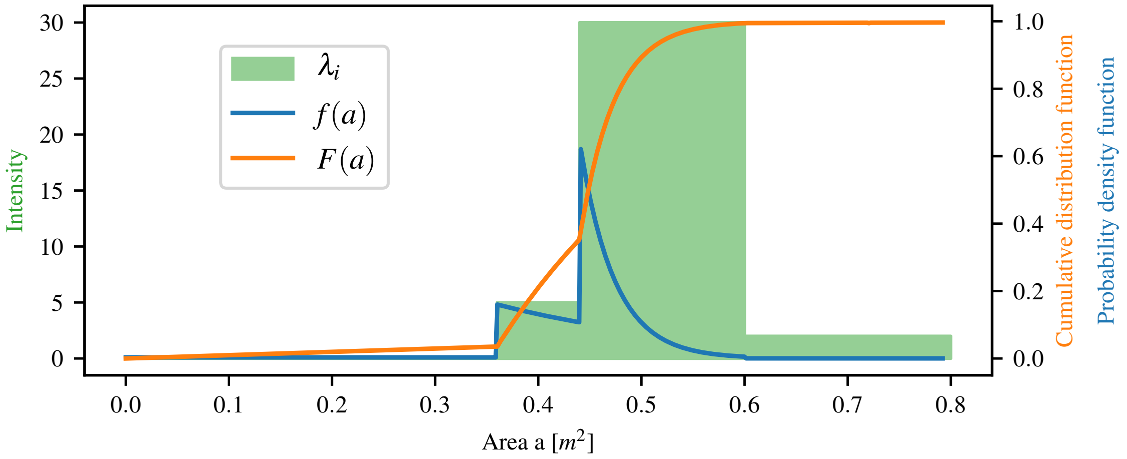


Figure 5.7: Example of lambda-Field the robot crosses (in green), with the associated probability density $f(a)$ (in blue) and cumulative distribution $F(a)$ (in orange).

The risk function $r(\cdot)$ is generic and can take into account the state of the robot as well as the state of the world. One can notice that the special case $r(\cdot) = 1$ leads to the probability of collision given by Equation 5.18. Furthermore, the probability density $f(a)$ only looks at the risk generated by the first collision occurring on the path. Therefore, it is assumed that the robot stops after any collision and does not continue its course. This assumption can be lifted if necessary, as shown in the next section.

For our applications, we chose to model the risk as the force of collision (i.e., loss of momentum) if the collision occurred at the area a. It is indeed a good quantification of the damage induced by the collision and is a better metric of the risk than the probability of collision, as shown by Eggert [109]. First, we present as an example a way to assess this risk assuming that every obstacle has an infinite mass. Indeed, this assumption holds for most scenarios where the robot's mass is negligible compared to the obstacles masses (e.g., a tree or a wall). We then remove this assumption in subsection 5.3.4 where each obstacle now has a probabilistic mass, allowing the robot to evolve in unstructured environments. One can note that other quantifications of the risk are also worth exploring, such as the loss of kinetic energy. Such a metric will be detailed in Chapter 6.

Assuming the obstacle that the robot collides with has an infinite mass, the force of collision is computed as

$$r(a) = m_R \cdot v_R^n(a), \tag{5.35}$$

where m_R is the mass of the robot, and $v_R^n(a)$ is its velocity towards the obstacle at the area a. As shown in Figure 5.8, the velocity towards the obstacle of normalized normal $\mathbf{n}$ is

$$\begin{aligned} v_R^n &= \left|\mathbf{n}^\top \mathbf{v_R}\right| \\ &= \left|v_R \cdot \cos(\theta)\right| \qquad \text{for } ||\mathbf{n}|| = 1 \end{aligned} \tag{5.36}$$

where $\cdot^\top$ stands for the usual vector transpose, $v_R = ||\mathbf{v_R}||$ the robot velocity and θ the angle between the robot heading and the obstacle's normal. The angle of collision is interesting to take into account for numerous scenarios, as for example an autonomous vehicle driving over a cliff. Because of skidding, the vehicle may find itself in a configuration where it has no choice but to collide with the safety railing. The best choice will naturally be to minimize the collision, hence collide with the railing with a high incidence angle.

This risk metric assumes that every obstacle the robot might encounter has an infinite mass. We also assume perfect inelastic collision, as most deployed vehicles are designed to absorb collisions as much as possible. It means that if the robot collides with an obstacle, the resulting collision would lead the robot to stop (i.e., losing a momentum of $m_R \cdot v_R^n$). Depending on the application, other metrics can be developed. We present in the next section the development of a more complicated metric allow-

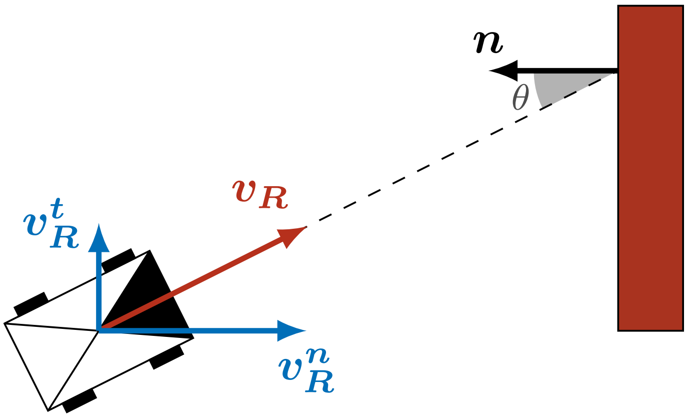


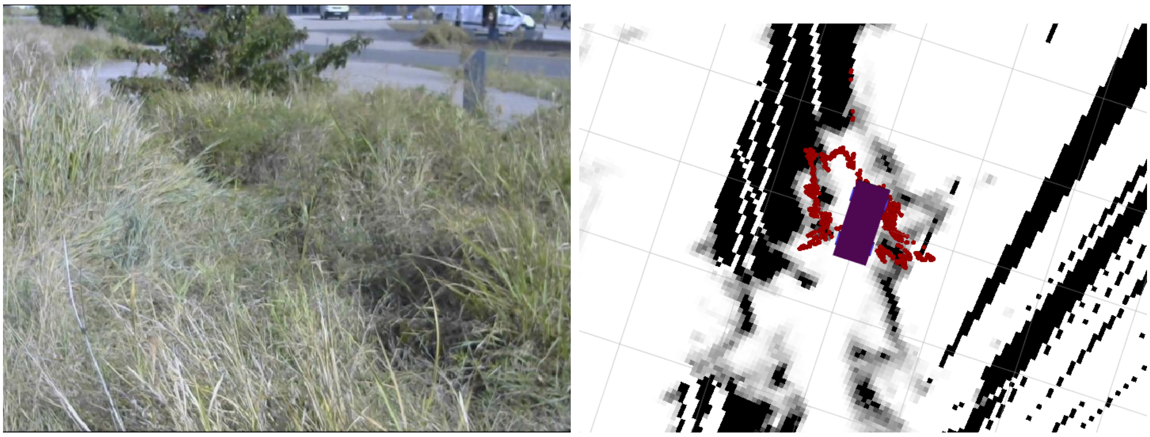
Figure 5.8: The robot of speed $v_R = ||\mathbf{v_R}||$ collides with an obstacle of normal $\mathbf{n}$ with an angle θ. The speed of the robot can be decomposed into the tangential component $\mathbf{v_R^t}$ and the normal component $\mathbf{v_R^n}$. Only the latter influences the collision with the obstacle.

ing the robot to navigate through unstructured obstacles like tall grass by lifting the approximation that all obstacles have infinite masses.

5.3.4 Taking into account the mass of the obstacles

In the context of autonomous navigation, the robot might have to go through objects that look like obstacles from the point of view of the lidar, but are in fact harmless for the robot. An ideal example of this scenario is where the robot have to go through tall grass to reach its goal. Figure 5.9 shows the sensors measurements of a robot trying to go through tall grass. Since the lidar returns very close measurements around the robot, the robot would be unable to move. However, with the images provided by the camera, an algorithm could clearly detect that the obstacles are only tall grass, hence the robot should proceed and reach its goal.

As the risk metric developed in the previous section assumes that every obstacle has an infinite mass, it is unable to deal with such scenarios. This assumption is then removed and each obstacle now has a probabilistic mass. We thereby need to estimate the mass of the obstacles. It can be done with a camera and deep learning segmentation like [278] or radar classification as done by [38]. We also take into account that the mass of an obstacle is probabilistic. In addition to the Lambda-Field, we also store a map of the probability distribution function of the mass distribution for each cell, which is provided by one of the above cited methods. Furthermore, as collisions with low-mass obstacles do not pose a threat to the robot, the risk metric is defined as the force of collision with obstacles that will stop the robot, therefore discarding threat-less collisions.



(a) (b)

Figure 5.9: The robot crosses an area with tall grass. (a) Camera view of the robot. (b) Lambda-Field created from lidar measurements in red, with the robot in purple. The darker the cell, the higher the lambda is. Without mass estimation, the robot would not move as it is certain that a collision will happen.

Figure 5.10 shows examples of probability distribution function for several obstacles. The main use of a probabilistic formulation for the masses is to deal with the uncertainty of the labels. Indeed, the grass can easily hide a high-density obstacle like rocks. Moreover, the mass of the vegetation is very variable and the robot can expect a harmless collision as much as a harmful collision, while going through these kinds of obstacles. In the case where no label is available for a cell, the worst case is taken into account, meaning that the mass of the cell is set to infinity.

We chose to discretize the probability density function into a sum of Dirac impulsions $\delta(\cdot)$. The mass p.d.f $f_i^m(\cdot)$ of the cell c_i is then

$$f_i^m(m) = \sum_{k=0}^{\infty} \alpha_{ik} \cdot \delta(m - k\Delta_m), \tag{5.37}$$

with Δ_m the discretization step and α_{ik} the probability that $m \in [k\Delta_m, (k+1)\Delta_m]$. Also, only a finite number of α_{ik} are not null in order to store the p.d.f.

A problem quickly arises from Equation 5.34 if we want to take into account the mass of the obstacles. The equation only looks at the first collision as it assumes that any collision would lead the robot to stop its course. For very light obstacles like grass, this assumption falls apart. Hence, we need to add a term to the equation to allow the robot to continue its course after a collision. To do so, we need to understand the meaning of the lambdas. For an area Δa where the Lambda-Field is constant with a value λ, the expected number of event 'collision' is $\Delta a \lambda$. Using the probability p_{si}, the probability of the robot being stopped because of the collision at the cell c_i, we want that each collision has the probability p_{si} of being harmful for the robot. Hence, we use our probability of traversal p_{si} as a new measure over the field. Given the harmful

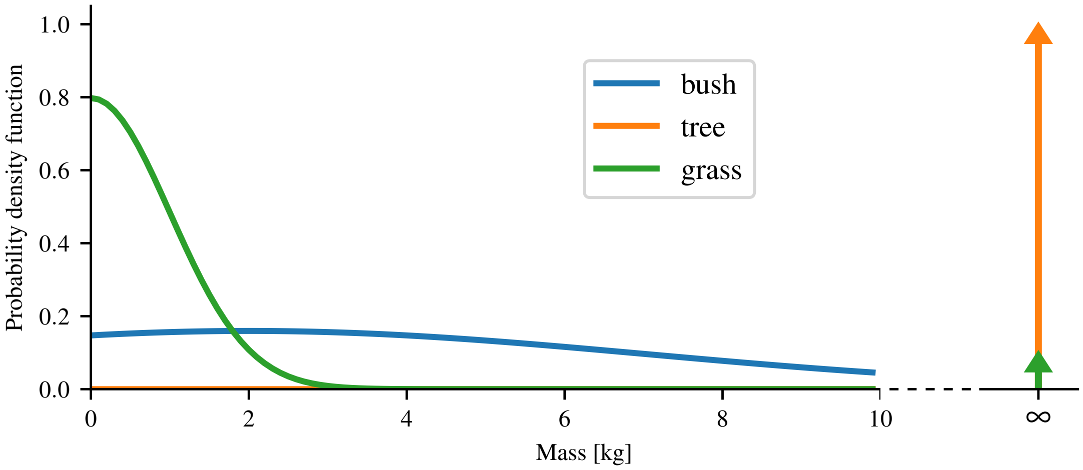


Figure 5.10: Examples of probability density function of several labels. The arrow represents the Dirac delta function. The mass of the grass is very likely to be close to zero but there is a chance that a high-mass obstacle is hiding in it (e.g., a rock). The mass of a bush is very uncertain as it may be more or less dense. In contrast, the mass of a tree is always very high.

probability p_{si} for the cell c_i, the intensity function $\Lambda(\mathscr{C})$ becomes

$$\Lambda_m(\mathscr{C}) = \Delta a \sum_{c_i \in \mathscr{C}} \lambda_i p_{si}. \tag{5.38}$$

Using this newly defined intensity measure, only hazardous collisions are investigated, treating collisions that do not stop the robot as harmless.

Assuming that the robot can go through obstacles if their mass is below a certain threshold $m_{\mathtt{max}}$, the probability p_{si} is then

$$\begin{aligned} p_{si} &= \mathbb{P}\left(m_i > m_{\mathtt{max}}\right) \\ &= 1 - \int_0^{m_{\mathtt{max}}} f_i^m(m)\,\mathrm{d}m, \end{aligned} \tag{5.39}$$

where $f_i^m(\cdot)$ is the p.d.f of the mass of the cell c_i. Also, since we do not assume anymore that the obstacles have infinite masses, the risk $r(\cdot)$ (i.e., the loss of momentum at the impact) becomes

$$\begin{aligned} r_m(\Delta ai, m) &= m_R\left(v_R^n(\Delta ai) - \frac{m_R v_R^n(\Delta ai)}{m_R + m}\right) \\ &= m_R \frac{m \cdot v_R^n(\Delta ai)}{m_R + m}, \end{aligned} \tag{5.40}$$

where m is the mass of the ith cell. Since the mass of each obstacle is probabilistic, we need to sum over all the possible masses to find the expected force of collision over a path, leading to (the proof is detailed in Appendix B):

$$\begin{aligned}\mathbb{E}[r_m(A,M)] &= \sum_{i=0}^{N-1} K_i \int_0^{\infty} f_i^m(m) r_m(\Delta ai, m)\,\mathrm{d}m \\ &= \sum_{i=0}^{N-1} K_i \sum_{k=0}^{\infty} \alpha_{ik} r_m(\Delta ai, k\Delta_m).\end{aligned} \tag{5.41}$$

where M is the random variable corresponding to the mass of the cell the collision happened, and K_i is computed the same way as in Equation 5.34 but using $\Lambda_m(\cdot)$. One can note that we can re-write the above equation under the form

$$\begin{aligned}\mathbb{E}[r_m(A,M)] &= \sum_{i=0}^{N-1} K_i\, r'(\Delta ai) \\ \text{with} \quad r'(\Delta ai) &= \sum_{k=0}^{\infty} \alpha_{ik} r_m(\Delta ai, k\Delta_m),\end{aligned} \tag{5.42}$$

hence going back to the known expectation formula of Equation 5.34. We omitted the parameters α_{ik} in the parameters of $r'(\cdot)$ as they are directly retrievable from the crossed area Δai. Also, one can note that setting $\mathbb{P}(m_i = \infty) = 1$ for all the cells leads as expected to the same risk as using Equation 5.35.

Using this metric, a robot can choose to take a path that leads to collisions but is harmless for it. However, it will be the same for the robot to take a path without any collision or a path with harmless collisions that do not stop the robot. For manned vehicles, a path with collisions will always be more uncomfortable for the user. Hence, we define a metric called the *mean risk* over a path. This quantity is fitted to choose between safe paths, as the mean risk cannot assure the safety of a path. Indeed, for a path with a lot of harmless collisions and a harmful collision, the mean risk will be very close to the harmless collisions because of the quantity disparity. The mean risk is defined as the mean collision the robot will undergo, computed as

$$\bar{r} = \sum_{i=0}^{N-1} \frac{\Delta a \lambda_i r(\Delta ai)}{\sum_{j=0}^{N} \Delta a \lambda_j}. \tag{5.43}$$

As the lambdas can be very large, a more convenient form can be found:

$$\overline{r} = \sum_{i=0}^{N-1} \frac{r(\Delta ai)}{\sum_{j=0}^{N} \lambda_j \lambda_i^{-1}} \tag{5.44}$$

where $\lim_{(\lambda_i,\lambda_j)\to(\infty,\infty)} \lambda_j \lambda_i^{-1} = 1$.

5.3.5 Comparison and improvement of the reachability metric

In this section, we analyze and adapt the concept of *reachability* defined in [267]. Their work is indeed the first to address the problem of risk assessment in occupancy grids. We first investigate the different metrics proposed in the article and then show that under our improvement to take into account the size of the robot, our framework can be seen as a generalization of their method. They propose to use the concept of product integrals, which is the product counterpart of the standard integration. A summary of the product integration can be found in [276]. They introduce the probability of occupancy $p_o(\cdot)$ (defined in their article as $m(\cdot)$) as the density of the cell. At first, they define the reachability R_t for a path from the time $t = 0$ to T as a product integral, computed as

$$R_t = \prod_0^T (1 - p_o(x(t)))^{\mathrm{d}t}, \tag{5.45}$$

where $x(t)$ is the robot position at the time t and $p_o(x(t))$ the probability that the position $x(t)$ is occupied. The higher the reachability, the safer is the corresponding path. However, they say that it would be better to consider the distance traveled through a cell instead of the time. It is indeed better as the first metric leads to a counter-intuitive reachability: for a robot crossing at a speed v a straight path of length l, where all cells have the probability p_o of being occupied, the reachability is

$$\begin{aligned} R_t &= \prod_0^{l/v} (1 - p_o)^{\mathrm{d}t} \\ &= \lim_{\Delta t \to 0} \prod_{i=0}^{l/v/\Delta t} (1 - p_o)^{\Delta t}. \end{aligned} \tag{5.46}$$

Using the fact that $(1 - p_o)^{\Delta t} = \exp\left(\ln(1 - p_o)\Delta t\right)$ and the Riemann definition of the integral, the expression can be simplified to

$$\begin{aligned} R_t &= \exp\left(\int_0^{l/v} \ln\left(1 - p_o\right) \mathrm{d}t\right) \\ &= (1 - p_o)^{l/v}. \end{aligned} \tag{5.47}$$

The reachability from the first metric is then higher when the speed is high, meaning that it is safer to travel the path at higher speed.

Their second reachability metric R_L does not possess such a behavior as they parametrized the integral over the traveled distance $L(t, t + \mathrm{d}t)$ between two instants, leading to

$$\begin{aligned} R_L &= \prod_0^T (1 - p_o(x(t)))^{L(t,t+\mathrm{d}t)} \\ &= \prod_0^T (1 - p_o(x(t)))^{|\dot{x}(t)|\,\mathrm{d}t}. \end{aligned} \tag{5.48}$$

Since the traveled distance $d(t)$ equals to $\int_0^t |\dot{x}(t)|\,\mathrm{d}t$, we have $\mathrm{d}d(t) = |\dot{x}(t)|\,\mathrm{d}t$ and Equation 5.48 can be simplified to

$$\begin{aligned} R_L &= \prod_0^D (1 - p_o(x_d(d))))^{\mathrm{d}d} \\ &= (1 - p_o)^D \quad \text{in case of homogeneous field,} \end{aligned} \tag{5.49}$$

where $D = \int_0^T |\dot{x}(t)|\,\mathrm{d}t$ is the total distance crossed by the robot and $x_d(\cdot)$ the position of the robot as a function of the traveled distance. Using Equation 5.49, the probability of collision does not depend on the tessellation size nor the speed of the vehicle. The main drawback is that there is no natural reason to use the concept of product integrals, as it is here merely a tool to make the probability constant. Furthermore, the robot is considered to be reduced to a point. The well-known solution to this problem is to inflate the obstacles, at the cost of assuming that the robot is round. As the Lambda-Field takes into account the size of the robot, we propose to improve their theory to take into account the robot's width W. Instead of only integrating over the robot line path, we also integrate over the entire width of the robot for each position $x_d(d)$. Under this consideration, the reachability equation becomes

$$R_L = \prod_0^D \prod_{-W/2}^{W/2} \left(1 - p_o(x(d, w))\right)^{\mathrm{d}w\,\mathrm{d}d}, \tag{5.50}$$

where $x(d, w)$ is a point of the robot parametrized as the distance the robot has traveled d and the distance from the robot header center in its width direction w.

We can see that for the special case $W = 1$ and $p_o(x(d, w))$ constant for $w \in [-W/2, W/2]$ we fall back on Equation 5.49. Assuming that the robot fully crosses the cells it encounters, we can develop a more convenient formulation for calculations. If the robot crosses the N cells $\mathcal{C} = \{c_i\}$ of size $\mathrm{S} \times \mathrm{S}\,\mathrm{m}^2$ and probability of occupancy

p_{oi}, Equation 5.50 can be re-arranged to give

$$\begin{aligned} R_L &= \prod_{i=0}^{N-1} \prod_{x=0}^{S} \prod_{y=0}^{S} (1 - p_{oi})^{\mathrm{d}x\,\mathrm{d}y} \\ &= \prod_{i=0}^{N-1} (1 - p_{oi})^{\Delta a}, \end{aligned} \tag{5.51}$$

where $\Delta a = S^2$ is the area of each cell.

From there, this equation can be linked to the theory of Lambda-Field. Indeed, for a path crossing the cells $\mathscr{C} = \{c_i\}$, in the Lambda-Field we have the probability of not colliding during the traversal computed as

$$\begin{aligned} 1 - \mathbb{P}(\texttt{coll}) &= \exp\left(-\Delta a \sum_{c_i \in \mathscr{C}} \lambda_i \right) \\ &= \prod_{c_i \in \mathscr{C}} \exp\left(-\lambda_i\right)^{\Delta a} \\ &= \prod_{c_i \in \mathscr{C}} (1 - p_{oi})^{\Delta a} \text{ with } p_{oi} = 1 - \exp\left(-\lambda_i\right). \end{aligned} \tag{5.52}$$

Hence, the probability of occupancy m_i of a cell is the probability of colliding in an area of $1\,\mathrm{m}^2$ in a Lambda-Field. Therefore, under our improvement given by Equation 5.51, the theory of [267] is then a special case of our framework, where the risk function $r(\cdot)$ is set to 1 and the cell size is assumed to be equal to 1. Compared to [267], we propose a more meaningful approach where the theory naturally gives a way to assess risk that is not restricted to be the probability of collision.

5.4 Experimentations

In this section, we present the experiments that validate the presented framework. First, we show the experimental setup that was used in the experiments. Then, we discuss the differences between the Lambda-Fields and the classical Bayesian occupancy grid. Finally, we provide experiments of mapping and planning showing the behavior of the robot in structured and unstructured environments.

5.4.1 Setup

We implemented our framework into a robot equipped with a LMS151 lidar and a camera, as shown in Figure 5.11. Since the robot has four-wheel steering, it was not much impacted by slipping and skidding and the odometry was sufficient to estimate the robot displacements. For every new lidar scan, the displacement between the current and previous position is estimated and the map is updated. Furthermore, the map is centered on the robot.

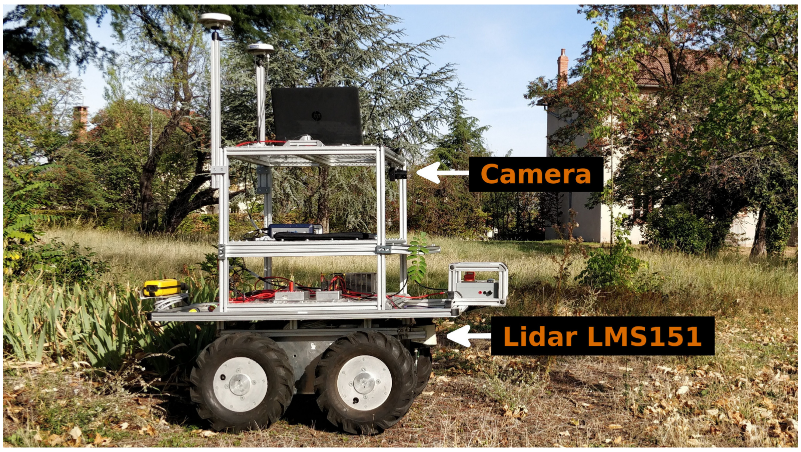


Figure 5.11: Robot used in the experimentations. It is equipped with a lidar Sick LMS151 and a camera.

We chose not to rotate the map but to rotate the robot instead to nullify the errors coming from the rotation. Indeed, a straight wall is quickly distorted after a few rotations because of the tessellation of the map. We also had to keep a global offset of the map as otherwise small displacements were not taken into account. Indeed, without this offset the map is not precisely updated as any displacement below half the cell size is discarded. The mass of the robot is also set to $m_R = 50\,\mathrm{kg}$. For safety purpose, the maximum speed of the robot was set to $0.5\,\mathrm{m\,s^{-1}}$ and the maximum acceleration to $0.05\,\mathrm{m\,s^{-2}}$. The parameters used for the confidence intervals of the mapping were $p^m = 0.9999$ and $p^h = 0.99$ for all cells measurements, where the cells have a size of $0.1 \times 0.1\,\mathrm{m}$. The value of p^m is intentionally very high as it is indeed nearly impossible for a lidar beam to go through obstacles. The normals of the obstacles were estimated using the method developed in [279]. At each lidar's scan, the normal of the points are estimated and the normals of each underlying cell c_i is updated as follows:

$$\bar{\theta} = \begin{cases} \arctan\left(\bar{S}/\bar{C}\right) & \text{if } \bar{C} \geq 0 \\ \arctan\left(\bar{S}/\bar{C}\right) + \pi & \text{otherwise,} \end{cases} \tag{5.53}$$

with

$$\bar{C} = \sum_{k=1}^{N} \cos(\theta_k) \quad \text{and} \quad \bar{S} = \sum_{k=1}^{N} \sin(\theta_k), \tag{5.54}$$

where N is the number of normal measurements θ_k for the cell c_i.

5.4.2 Comparison with the Bayesian occupancy grid

In order to demonstrate the difference between the Bayesian occupancy grid and the Lambda-Fields, we theoretically investigate the key differences between the two frameworks, then show on real-world experiments their consequences on the quality of the maps.

First, we investigate the convergence of the occupancy of a single cell. In the context of unstructured environment, it is very common to have cells that are only partially occupied. This can come from either very thin objects such as tall grass or crops, or from obstacles that do not reflect well the laser beam, such as a dense bush. In both cases, the cell will be measured both 'hit' and 'miss' as the beam can cross or hit the obstacles in the cell. Using the theory presented in [1] to construct the Bayesian occupancy grid, we use the log odds representation of occupancy. Assuming that a cell is measured N times and is filled at a ratio of $r \in [0,1]$ (1 is completely filled, 0 is completely empty), the cell will be measured 'hit' rN times and 'miss' $(1-r)N$ times. The Bayesian occupancy grid estimates the occupancy probability of the cell as

$$\mathbb{P}\left(\mathtt{occ}_b\right) = 1 - \frac{1}{1+\exp\left(rNl_o + (1-r)Nl_f\right)}, \tag{5.55}$$

where l_o is the log odds representation of the probability that the cell is occupied given a 'hit' measurement, whereas l_f is the log odds representation that the cell is occupied given a 'miss' measurement (i.e., informs that the cell is free). These quantities are computed as

$$\begin{aligned} l_o &= \ln\left(\frac{\mathbb{P}\left(\mathtt{occ}|z=\mathtt{hit}\right)}{1-\mathbb{P}\left(\mathtt{occ}|z=\mathtt{hit}\right)}\right), \text{ and} \\ l_f &= \ln\left(\frac{\mathbb{P}\left(\mathtt{occ}|z=\mathtt{miss}\right)}{1-\mathbb{P}\left(\mathtt{occ}|z=\mathtt{miss}\right)}\right), \end{aligned} \tag{5.56}$$

with z the measurement of the sensor that is either 'hit' or 'miss'. Substituting the definition of the log odds representation with its expression, we have

$$\begin{aligned} \mathbb{P}\left(\mathtt{occ}_b\right) &= 1 - \frac{1}{1+\left[\left(\frac{\mathbb{P}(\mathtt{occ}|z=\mathtt{hit})}{1-\mathbb{P}(\mathtt{occ}|z=\mathtt{hit})}\right)^r\left(\frac{\mathbb{P}(\mathtt{occ}|z=\mathtt{miss})}{1-\mathbb{P}(\mathtt{occ}|z=\mathtt{miss})}\right)^{1-r}\right]^N} \\ &= 1 - \frac{1}{1+\left[O_o^r \cdot O_f^{1-r}\right]^N}, \end{aligned} \tag{5.57}$$

where $O_o, O_f \in \mathbb{R}_{\geq 0}$ are defined as the odds of respectively $\mathbb{P}\left(\mathtt{occ}|z = \mathtt{hit}\right)$ and $\mathbb{P}\left(\mathtt{occ}|z = \mathtt{miss}\right)$. Taking the limit $N \to \infty$, we have

$$\lim_{N\to\infty} \mathbb{P}\left(\mathtt{occ}_b\right) = \begin{cases} 1 & \text{if } O_o^r \cdot O_f^{1-r} < 1 \\ 0.5 & \text{if } O_o^r \cdot O_f^{1-r} = 1 \\ 0 & \text{if } O_o^r \cdot O_f^{1-r} > 1. \end{cases} \tag{5.58}$$

Therefore, we see that the Bayesian occupancy grid will always converge to an extremum (apart from the special case where $O_o^r \cdot O_f^{1-r} = 1$, meaning that the measurements do not provide information about the occupancy). On the contrary, the Lambda-Field does not converge to an extremum. Indeed, putting the estimation of lambda given by Equation 5.23 into the probability of collision of Equation 5.17, we have

$$\begin{aligned} \mathbb{P}\left(\mathtt{occ}_\lambda\right) &= 1 - \exp\left(-\Delta a \lambda\right) \\ &= 1 - \exp\left(-\Delta a \cdot \frac{1}{e} \ln\left(1 + \frac{Nr}{N(1-r)}\right)\right) \\ &= 1 - \left(1 + \frac{r}{1-r}\right)^{-\frac{\Delta a}{e}}. \end{aligned} \tag{5.59}$$

In the case where the lidar error region e is equal to the area of the cells Δa, meaning that we are sure that the collision comes from this cell, the equation simplifies to

$$\begin{aligned} \mathbb{P}\left(\mathtt{occ}_\lambda\right) &= 1 - \frac{1}{1 + \frac{r}{1-r}} \\ &= r, \end{aligned} \tag{5.60}$$

meaning that the probability of collision is purely the ratio of occupancy of the cell and does not depend on the number of measurement N. Figure 5.12 shows the convergence of the Bayesian occupancy grids and the Lambda-Field. In order to ease the reading, the amount of information a 'hit' measurement is the same as for a 'miss' measurement of the cell, meaning that $\mathbb{P}\left(\mathtt{occ}|z = \mathtt{hit}\right) = 1 - \mathbb{P}\left(\mathtt{occ}|z = \mathtt{miss}\right)$ and thus $l_o = -l_f$. The behavior still remains the same without this assumption as shown by Equation 5.58. The Lambda-Field estimation does not move and stays at the true occupancy value of the cell, whereas the occupancy probability of the Bayesian occupancy grid converges to either 0 or 1 depending on the occupancy. As expected, the higher the confidence of the sensor's measurement, the faster the probability converges. This behavior can be hazardous in unstructured environment, as a cell filled at 49 % will converge to a probability of occupancy of 0 after a few seconds.

We also consider the case where an obstacle is wrongly undetected. This situation can happen when measuring sparse or unstructured obstacles. For instance, a wire fence can be either stop or let through the laser beams depending on the position of

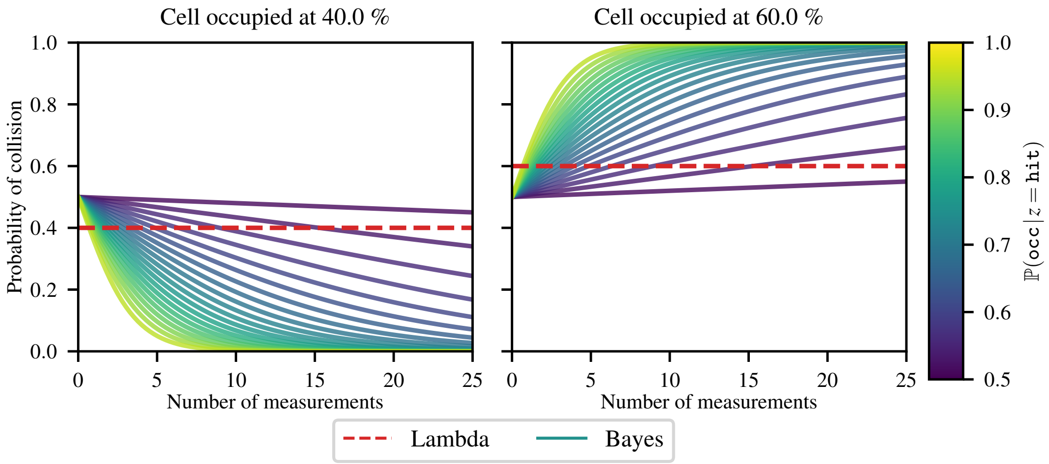


Figure 5.12: Convergence of the collision probability for a given cell occupied at 40 % and 60 %. The Bayesian occupancy grid will quickly converge to a probability of occupancy of either 0 or 1 whereas our framework stays at the true occupancy value of the cell.

the robot. Thus, we look at the speed at which the Bayesian occupancy grid and the Lambda-Field can recover the true state of an obstacle wrongly labeled as free. For a single cell measured m times as 'miss', we assume that m is large enough such that the Bayesian occupancy grids converged to the probability of occupancy $\mathbb{P}(\texttt{occ}) = 0$. Then, assuming that after m measurements, the robot starts to measure the obstacle as 'hit', we can approximate the probability of occupancy around $h \approx 0$ using first order Taylor expansion as

$$\begin{aligned}\mathbb{P}\left(\texttt{occ}_\texttt{b}\right) &= 1 - \frac{1}{1 + \exp\left(ml_f + hl_o\right)} \\ &\approx \frac{l_o \exp\left(ml_f\right)}{\left(\exp\left(ml_f\right) + 1\right)^2} \cdot h \\ &\approx \frac{l_o}{2\left(\cosh\left(ml_f\right) + 1\right)} \cdot h,\end{aligned} \tag{5.61}$$

meaning that the recovery rate of the Bayesian occupancy grid vanishes exponentially as a function of the number of times m the cell has been wrongfully measured. In the case of the Lambda-Field, we have

$$\begin{aligned}\mathbb{P}\left(\texttt{occ}_\lambda\right) &= 1 - \exp\left(-\frac{\Delta a}{e} \ln\left(1 + \frac{h}{m}\right)\right) \\ &= 1 - \frac{1}{(1 + h/m)^{\Delta a/e}} \\ &\approx \frac{\Delta a/e}{m} \cdot h,\end{aligned} \tag{5.62}$$

indicating that the recovery rate vanishes linearly as a function of the number of times m the cell has been wrongfully measured. Figure 5.13 shows the convergence curves towards the true state of the cell (i.e., 100 % filled) for different values of confidence of the sensor measurements.

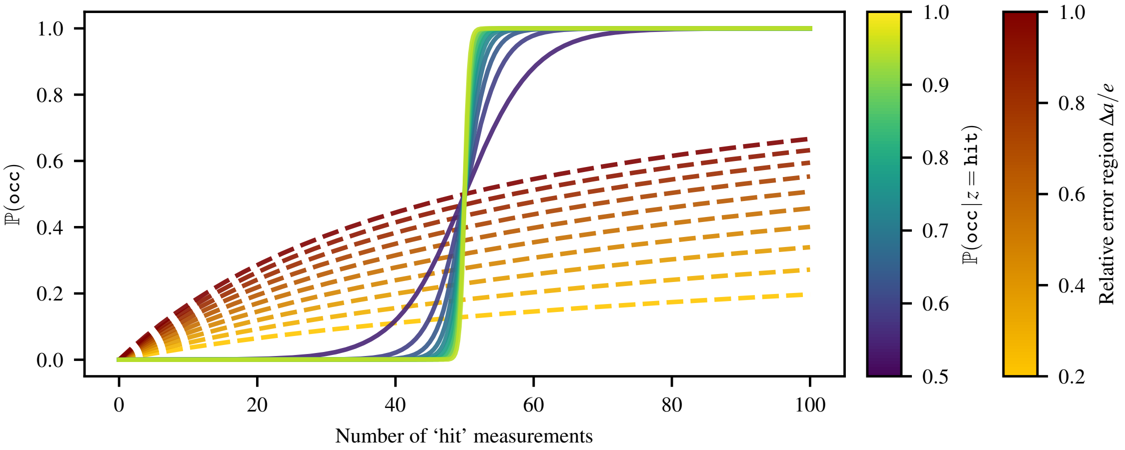


Figure 5.13: Evolution of the collision probability of a fully occupied cell that has been wrongly measured empty 50 times before. As theoretically shown, the Lambda-Field manages to recover quicker from the wrong estimation, whereas the Bayesian occupancy grid converges faster to the true state of the cell after 50 'hit' measurements.

The cell has been measured 50 times as 'miss' before (e.g., the robot stayed still during 2 s for a 25 Hz lidar such as the Sick LMS151). Then, the robot changed position, allowing the lidar beams to effectively hit the obstacle in the cell, leading to 'hit' measurements in the cell. As expected, the higher the confidence on the sensor (i.e., smaller error region for the Lambda-Field and higher probability for the Bayesian occupancy grid), the faster the recovery speed is. However, the Lambda-Field allows to better recover at the beginning by growing faster, whereas the Bayesian occupancy grid prefers to quickly converges to the state 'occupied' after 50 'hit' measurements. The Lambda-Field does however take more time to converge toward a full occupancy of the cell as it still takes into account the previous wrong 'miss' measurements. Indeed, as shown in Figure 5.12, the Bayesian occupancy grid only converges to zero or one. Therefore, as soon as the 'hit' measurements become predominant over the wrong 'miss' measurements, the framework quickly converges to 1. Both frameworks can have their convergence speed shortened by applying a threshold on the probability of occupancy and the lambda (i.e., cannot go above or below certain values). However, this enhancement does not modify the previous analysis as it only bounds the vanishing of the recovery rate.

In order to demonstrate in real-world conditions these considerations, we mapped a small zone consisting of a black wire fence where lidar beams could easily get through, as shown in Figure 5.14. At first, the position of the robot leads the laser beams to cross the fence without collision, yielding both the Bayesian occupancy grid and the Lambda-Field to converge to a false state. After a few seconds, the robot turns in front of the fence, leading more lasers beams to actually collide with the obstacle. In this configuration, the two aforementioned differences between the Lambda-Field and Bayesian occupancy grid are involved.

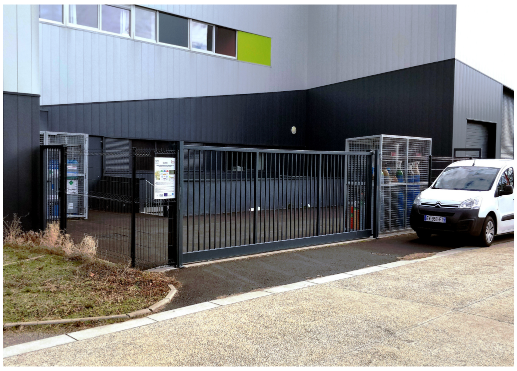


Figure 5.14: Mapping environment, consisting of a spare wire fence with some tall grass on the left, as well as a small truck.

On the one hand, the laser beams still have a chance to go through the fence, leading to cells that are not completely filled to wrongly converge for the Bayesian occupancy grid. If the lasers collide with the cell less than 50 % of the time, the Bayesian occupancy grid will wrongly converge to 0. On the other hand, because of the wrong 'miss' measurements at the beginning, the Bayesian occupancy grid will struggle more to recover the true state. Figure 5.15 provides the resulting Lambda-Field and Bayesian Occupancy at two different times during the mapping process. We investigated the quality of the mapping by quantitatively analyzing the ratio of wrongly undetected cells, as shown in Figure 5.16.

In order to measure the quality of the map, we manually labeled the fence at each iteration and used patches of 4 × 4 cells running along the fence. The inspected zone is shown in Figure 5.15 in dashed green. Using patches instead of directly analyzing the cells removes the noise due to the manual labeling and the noisy odometry of the robot. Using these patches, we evaluate the recall of the detection, computed as the sum of the probabilities of not colliding each patch over the number P of patches (i.e., the proportion of wrongly detected patches) as

$$\begin{aligned} \text{Recall}_b &= \frac{1}{P}\sum_{k=0}^{P-1}\prod_{c_i\in p_k}\left(1-\mathbb{P}\left(p_i=\texttt{occ}\right)\right), \\ \text{Recall}_\lambda &= \frac{1}{P}\sum_{k=0}^{P-1}\exp\left(-\Delta a\sum_{c_i\in p_k}\lambda_i\right). \end{aligned} \tag{5.63}$$

In addition to the recall that can be seen as the mean of the probabilities of not colliding the patches, we also compute the associated standard deviation of the patches. Figure 5.16 depicts the recall for the Lambda-Field and Bayesian grid as well as the distribution at two sigmas (approximately 95 %) of the probability of not colliding the patches.

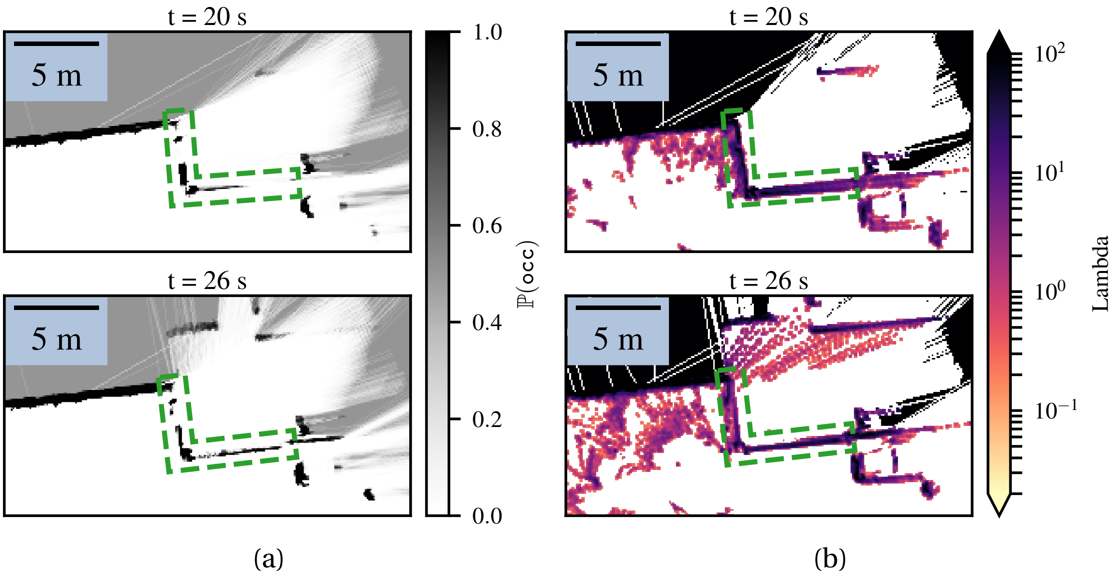


Figure 5.15: Mapping of a wire fence where, depending on the robot pose, the laser beams can either collide or go through the fence (outlined in dashed green in the maps). (a) Bayesian occupancy grid of the wire fence. (b) Lambda-Field of the wire fence. One can note that the Bayesian occupancy grid did not have the time to converge at $t = 20$ s because of the wrong 'miss' measurements at the beginning of the experiment.

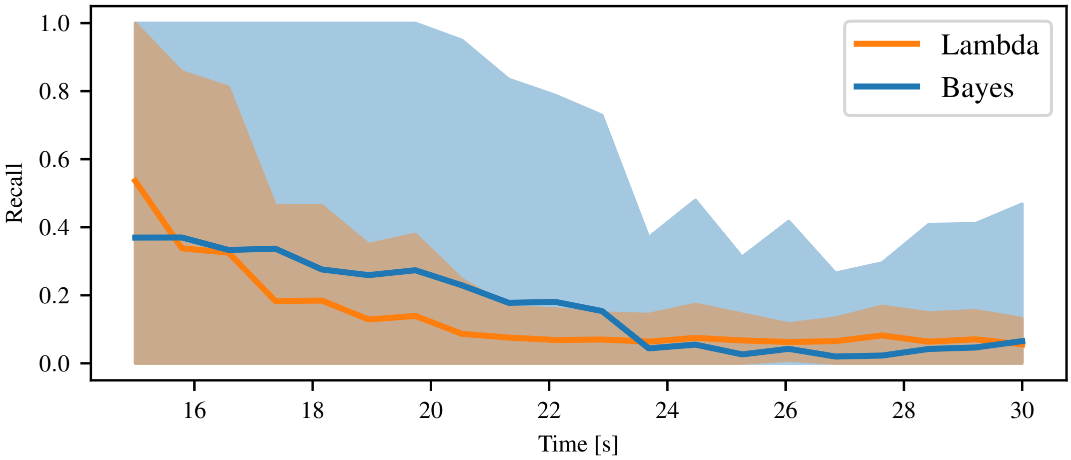


Figure 5.16: Mapping error of the wire fence for the Bayesian occupancy grid and the Lambda-Field. The error is defined as the ratio of free space over the whole space that represents the obstacle. As expected, the Lambda-Field converges quicker to a low error whereas the Bayesian occupancy grid needs more time to assess its occupancy. Although the Bayesian occupancy grid yield to a smaller error after $t = 24$ s, its confidence interval is twice as large as the one of the Lambda-Field, meaning that the representation of the wire fence is still sparse and contains large errors.

As theoretically expected, the Lambda-Field recovers quicker from the wrong 'free' state of the wire fence. At $t = 20\,\mathrm{s}$, the Lambda-Field converged to the state where the whole fence is considered as an obstacle, whereas the Bayesian occupancy grid is still converging and has more than 20 % of the fence that is considered as free. At $t \approx 22.5\,\mathrm{s}$, the robot changes position such that parts of the fence are not visible to the lidar sensor. As such, the Lambda-Field started to converge toward a lower lambda that is mainly seen on the left vertical wall. Although the recall is lower for the Bayesian occupancy grid, we can still see holes in the fence at $t = 26\,\mathrm{s}$ while the Lambda-Field has the whole fence mapped. Indeed, the distribution at 95 % of the probabilities of the patches for the Lambda-Field is almost always contained in the one of the Bayesian occupancy grid. This means that although the recall is lower, the bayesian occupancy grid has patches that are poorly represented compared to the Lambda-Field. One can also note that the car on the right of the map is better represented in the Lambda-Field than in the Bayesian occupancy grid.

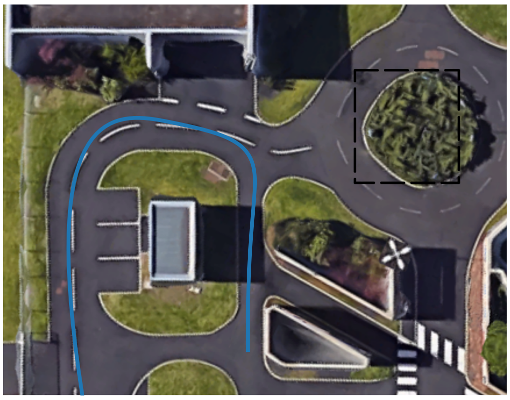


Figure 5.17: Aerial View of the mapped environment, with the robot path in blue and the roundabout in dashed black

Then, we show the effectiveness of our framework to map large environments. To do so, we implemented a simple robot follower scenario in an urban-like environment depicted in Figure 5.17. The robot had to follow the pedestrian while keeping the risk of the chosen path below $5\,\mathrm{kg\,m\,s^{-1}}$. While following the pedestrian, the robot created a Lambda-Field as well as a Bayesian occupancy grid of the environment, as shown in Figure 5.18. In addition to the expected lambdas, the Lambda-Field also stores a confidence interval for each value. Figure 5.19 shows the difference between the expected lambdas λ_i and the lower and upper values λ_L, λ_U of the confidence interval. It can be easily seen that the lower lambdas only underestimate the lambdas for the obstacles. Indeed, the expected lambda is already at the lowest possible value, zero, for the cells where no collision has happened. They can however be useful in the case where a lot of faulty measurements come from the sensor, for example when lidars try to map an environment under heavy rain or during a snowstorm. On the contrary, upper lambdas overestimate the lambdas for regions where few measurements are available. It is especially the case in the roundabout or in the boundary of the map, as well as behind the fence on the left side of the map where only a few lidar beams passed through.

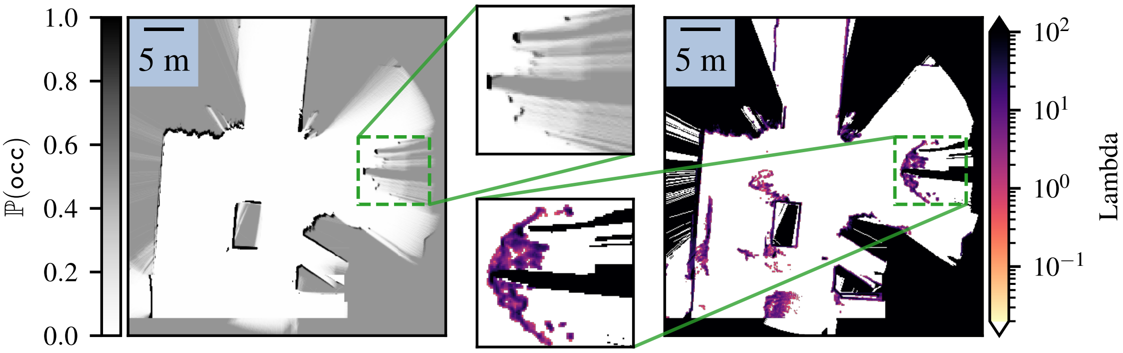


Figure 5.18: *Left:* Bayesian occupancy grid *Right:* Lambda-Field. The Lambda-Field is better suited to store the occupancy of unstructured obstacles where the Bayesian occupancy grid may over-converge, especially for the roundabout in dashed green.

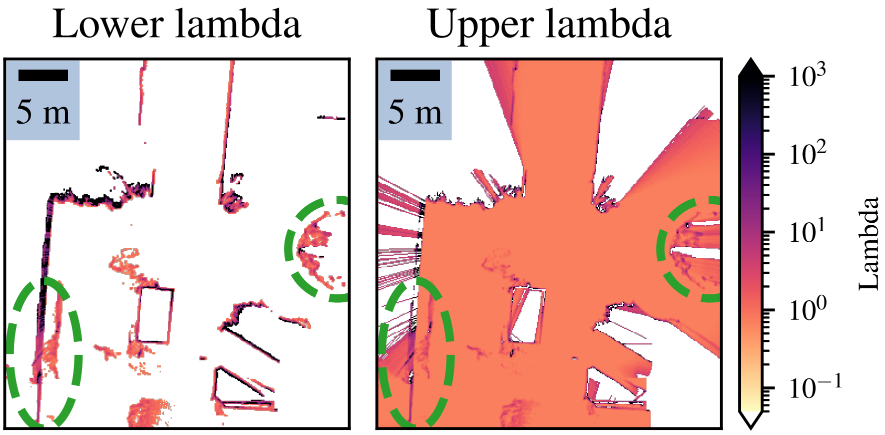


Figure 5.19: Difference of lambdas between the expected lambdas and the confidence interval for a structured environment. The lower lambdas tend to underestimate the obstacles as the expected lambdas in the free space are already at the lowest value. The upper lambdas overestimate the field everywhere, particularly where few measurements were available, like in the roundabout or behind the fence on the left (circled in green).

The maps are globally alike except for the unstructured obstacles, which are in this case the bushes in the roundabout as well as tall grass around the pavement. As the Bayesian occupancy grid needs to converge to either the state 'occupied' or 'free', a lot of information about the occupancy of the roundabout is discarded. Furthermore, the robot was not able to see the entire obstacle at first, leading the frameworks to wrongly converge. As shown in the previous section, the Lambda-Field recovers faster in these situations, leading to a more precise map. Most of the information is preserved in the Lambda-Field and the global shape of the roundabout is more easily recognizable. The other disparities between the two maps come from unstructured obstacles which were small trees and tall grass.

Finally, we also mapped an unstructured environment where the environment is depicted in Figure 5.20 and the resulting maps are shown in Figure 5.21. The environment consists of several trees with a lot of tall grass disrupting lidar measurements. The robot went around the tree in the center of the picture while navigating in the grass. However, due to the grass and the wind, the lidar returned a lot of measurements corresponding to the grass. Whereas the Bayesian occupancy grid only kept the hedge and the main trees, the Lambda-Field kept more information, such as the wooden benches on the top of the map or the tall grass. This behavior is caused by the necessity of the Bayesian occupancy grid to always converge to an extremum, leading it to discard a lot of information that can be critical. For instance, in the case of agricultural robotics, keeping the crops on the map is essential for not rolling over it.

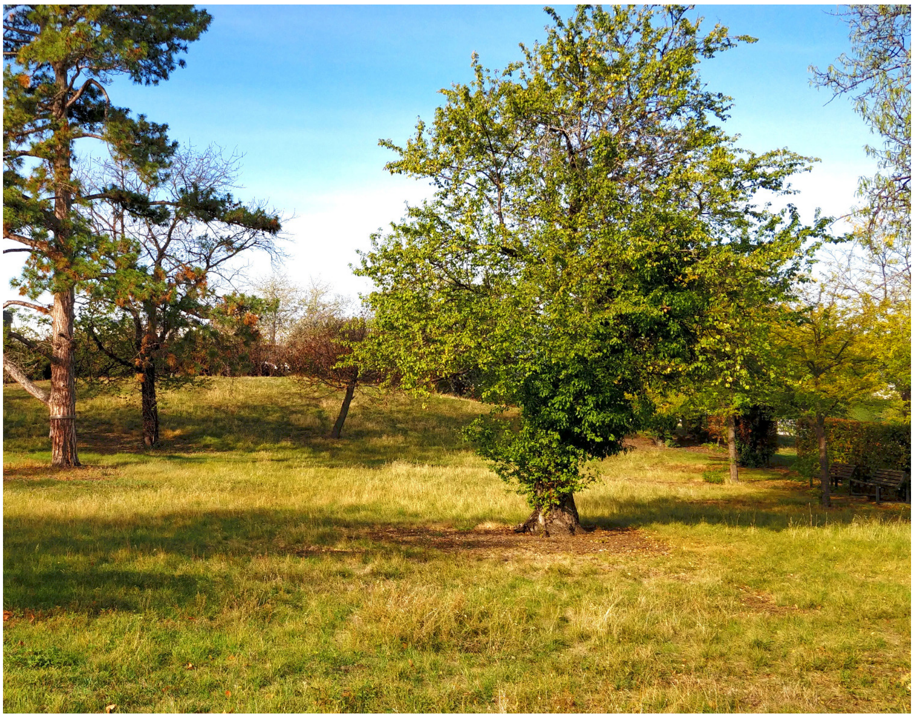
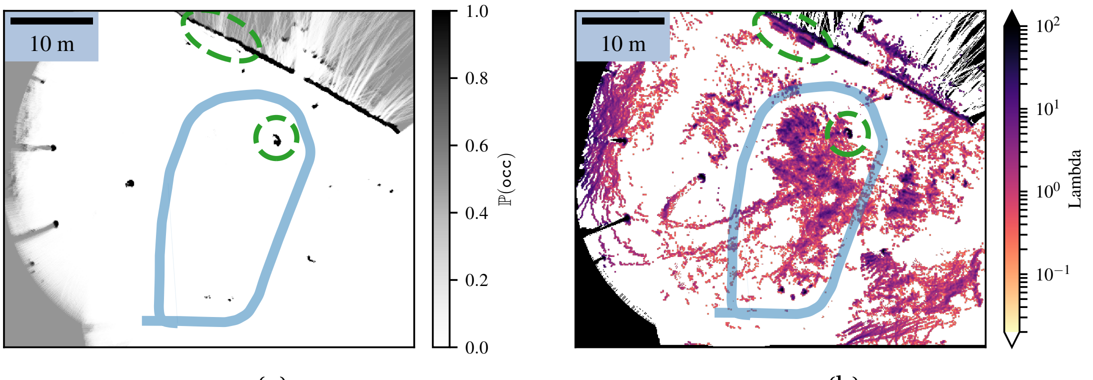


Figure 5.20: Unstructured environment used for the mapping experimentation, consisting of tall grass that disrupted lidar measurements, trees with large trunks, a hedge on the right of the picture and benches next to the hedge.



(a) (b)

Figure 5.21: Mapping of an unstructured zone. (a) Bayesian occupancy grid of the environment (b) Lambda-Field of the environment. The robot, with its path in light blue, went around the nearest tree (circled in green), before going back to its initial position. Because of the tall grass, the Lambda-Field stored a lot of obstacles during the traversal while the Bayesian occupancy grid discarded the vast majority of them. The hedge and the tree trunk (circled in green) are still visible in both maps as they are the only structured obstacles, whereas more unstructured obstacles such as benches on the top of the map (circled in green) or bushes have been discarded by the Bayesian occupancy grid.

5.4.3 Basic path planning

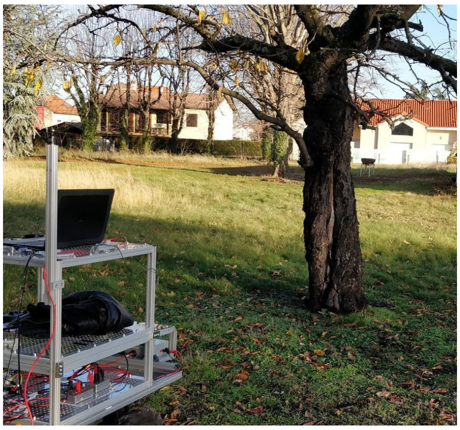
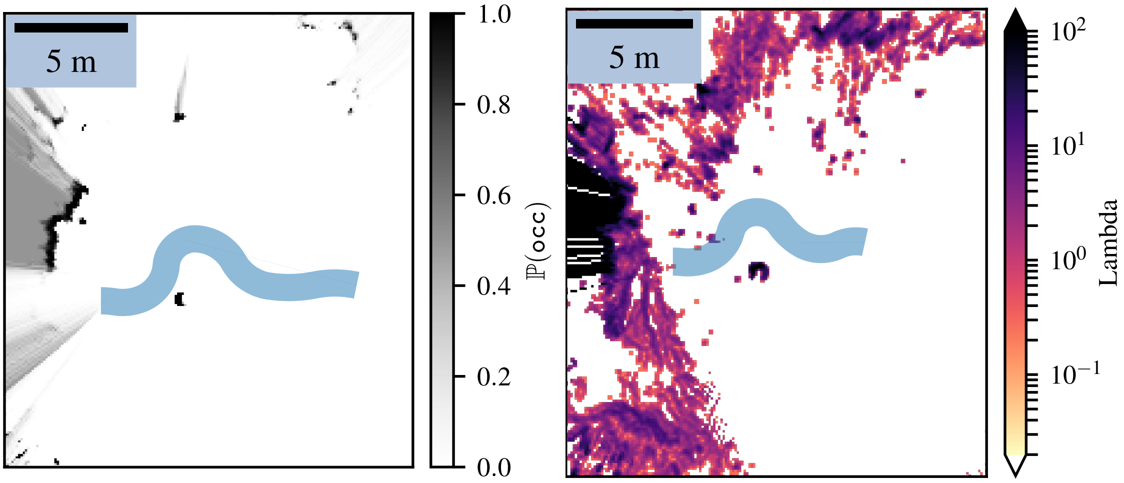


Figure 5.22: The robot has to avoid a tree which was on its path in order to reach the goal.

We demonstrate in this part that our framework can be used to do classical path planning. As shown in Figure 5.22, the robot has to go around a tree to reach the goal which was set behind. To do so, we implemented the path planning algorithm of [280]. Every 3 seconds, we sample feasible commands for the robot and choose the best one. The best command (i.e., path) is the one that stays below a risk threshold and leads the robot the closest possible to the target, which was in this case behind the tree. The chosen path also requires to have an upper risk (i.e., the risk computed using the upper bound of the lambdas) below another risk threshold. For each feasible command, the N cells crossing the path induced by the command applied for 8 seconds are extracted and the risks are computed. Estimating the risk of a longer time than the one applied by the command avoids the robot choosing paths leading to a dead end. Indeed, if the risk was computed only for the time of the command, the applied command might lead the robot to be right in front of a wall, resulting in a configuration where it is impossible to escape. In the case where no command meets the criteria, the robot stops. It can happen when the robot is at high-speed: because of the limited deceleration, all the high-speed commands lead to a risk higher than the maximum allowed. Then, the robot completely stops before continuing its course as it can now sample low speed commands. Although the path planning algorithm is somewhat basic, the aim is to show the applicability of the framework in real world scenarios.



(a)

(b)

Figure 5.23: (a) Bayesian occupancy grid with the path the robot has taken in light-blue. (b) Lambda-Field with the path the robot has taken in light-blue.

We also implemented the Reachability metric of [267] with our improvement to handle the robot size. As converting the Lambda-Field into an occupancy grid using Equation 5.52 would lead to computing the risk $r(\cdot) = 1$, hence using our theory, we used the Bayesian occupancy grid directly computed from lidar measurements. Using the same method for path planning, we sample paths and choose the one which leads the closest to the goal, where a path is considered safe if its reachability R_L is above a certain threshold $1-\epsilon, \epsilon \in]0,1[$. The threshold ϵ was set to 0.1 during our experiments. For each applied command, 300 samples are evaluated.

Using both algorithms on the same environment allows the comparison of the behaviors of the robot in a simple case. Figure 5.23 shows the result of the path planning for the two environment representations. For the Lambda-Field, the maximum allowed expected risk was set to $0\,\mathrm{kg\,m\,s^{-1}}$ meaning that the robot must remain clear of any collision. We see that the paths are much alike and the robot effectively avoids the obstacles in both cases. It can be seen on the Lambda-Field that the robot path crossed some cells where the lambda is not null, which would lead to a collision. However, as the Lambda-Field is computed in real-time, the lidar measured collisions in this cell after the robot crossed it. The lidar beams can indeed go through the grass or returns a collision depending on the position of the robot.

The same experiment was conducted using different parameters for the Lambda-Field. Figure 5.24 shows the resulting speed of the robot for different configurations of parameters. While the robot had to expect no risk on its path, it was first allowed to have an upper risk at $5\,\mathrm{kg\,m\,s^{-1}}$, meaning that we are sure at 95 % that any unexpected collision has an expected risk below $5\,\mathrm{kg\,m\,s^{-1}}$. The robot quickly reached its maximal speed with full acceleration while keeping the upper risk below the threshold. Under the same configuration, the robot had to reach the goal while keeping the upper risk below $2\,\mathrm{kg\,m\,s^{-1}}$. As the upper risk is smaller, the robot had to reduce its speed. The robot has the same type of reaction if its confidence in the lidar sensor decreases. In the third experiment, the probability of right 'miss' measurement p^m was decreased from 0.9999 to 0.99. The direct implication is the confidence interval broadened, forcing again the robot to decrease its speed. Then, the robot had a poor confidence in the lidar as well as a small upper risk, leading to a very slow traversal speed. Thereupon, the Lambda-Field allows classical path planning in the same fashion as [267]. Our framework also regulates the speed of the robot to cope with the allowed risk level.

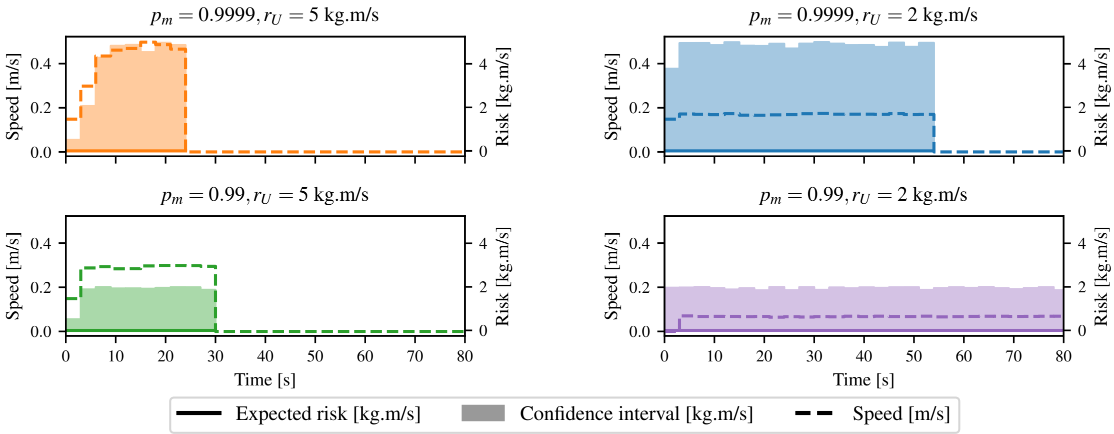


Figure 5.24: Speed (dashed line) and risk (solid line with shaded area for its confidence interval) of the chosen paths of the robot going around a tree for different configurations. The robot is able to navigate at a higher speed when it is confident about the measurements and has a higher upper risk limit.

5.4.4 Going through tall grass

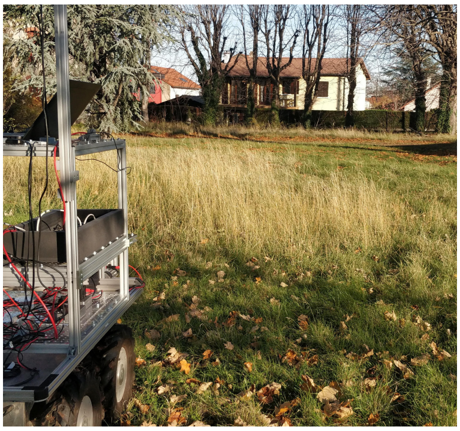


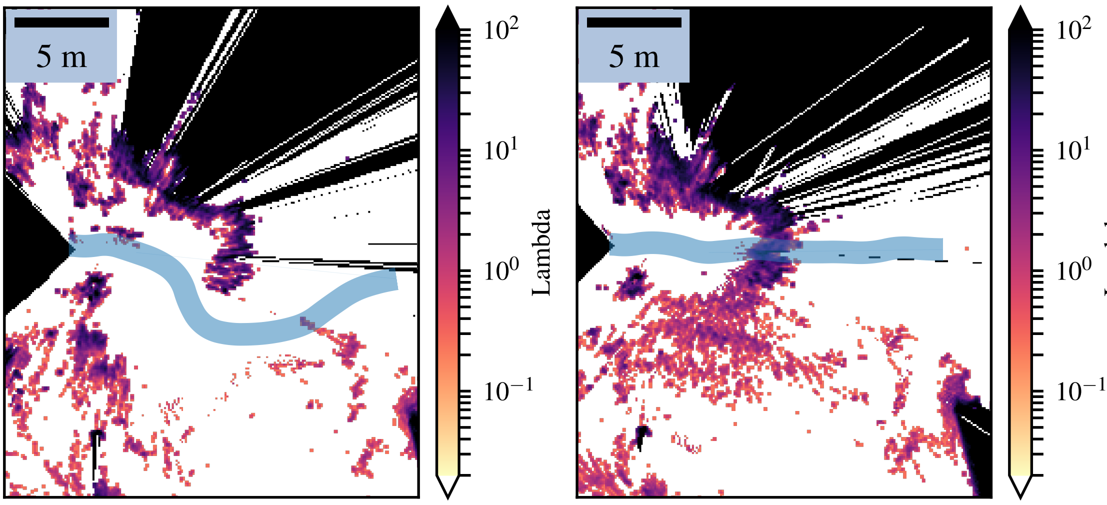
Figure 5.25: The robot has to reach a goal behind the tall grass.

After doing simple path planning, we show that the Lambda-Fields allow the robot to navigate in unstructured environments. As shown in Figure 5.25, the robot has to reach the goal which is behind tall grass. This kind of environment leads to very noisy maps which can hinder the robot's displacements if looking at the probability of collision. We here show that according to the risk the robot is willing to take, it chooses to either go through the tall grass or trying to find a path around it. This kind of behavior is impossible to have when only looking at the probability of collision. Indeed, the robot is sure to collide with the grass. The probability of collision is high but the collision caused by the grass is harmless, leading to a very small risk.

We assume that using a camera, the robot knows that the obstacles in front of it are tall grass. For any other zone, the obstacle masses are set to the worst case (i.e., $\mathbb{P}(m_i = \infty) = 1$). We assumed that tall grass has a 95 % chance to have a null mass and a 5 % chance of having an infinite mass. This probability models the possibility that tall grass can hide very dense obstacles like rocks or tree trunks.

Two cases are analyzed: in the first one, the robot had to take no risk, meaning that for every path the robot takes, the expectation of the risk has to be zero. Hence, the robot chose to go around the tall grass. In the second case, the robot was allowed to take some risk to reach its goal and went through the tall grass. Figure 5.26 shows the resulting Lambda Fields for these two different robot configurations.

In the first case 5.26 (a), since the tall grass has a non-zero probability to have a mass which leads to a harmful collision, the robot chooses to go around the tall grass to reach the goal. Once again, the cells with a lambda higher than zero on the path of the robot have been updated after the robot went through it. At the time the robot crossed these cells the lambdas were null. Figure 5.27 shows the speed as well as the risk taken by the robot. The robot first crosses a zone where the mass is supposed to be low, leading to a very narrow confidence interval. The confidence interval grows quickly as the robot goes out of the low mass zone. The robot also stopped several times during the traversal. Indeed, a lot of grass hindered its movements as the detection of the grass is very random while moving. Because of the maximum deceleration of the robot, no path were below the maximum risk allowed. The robot then had no choice but to completely stop to be able to plan with low speed commands.



(a)

(b)

Figure 5.26: (a) Lambda-Field with the path the robot has taken in blue, where the robot was instructed to take absolutely no risk. (b) Lambda-Field with the path the robot has taken in blue, where the robot was allowed to take some risk.

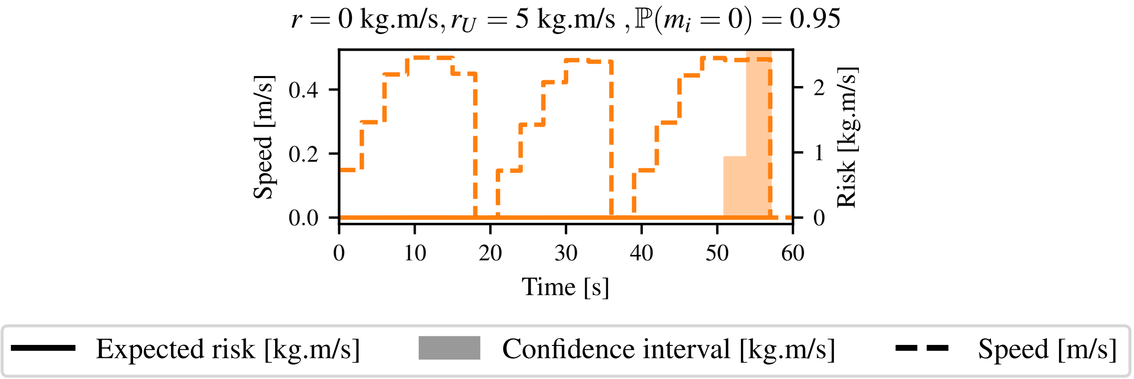


Figure 5.27: Speed (dashed line) and risk (solid line with shaded area for its confidence interval) of the chosen paths of the robot when it chooses to go around the tall grass. The numerous stops of the robot are due to the very random detection of the tall grass.

In the second case 5.26 (b), the robot was allowed to take some risk and chose to go through the tall grass to reach the goal. The differences of lambdas between the two maps come from the fact that depending on the robot position, the lidar beams can go through the grass or return a collision. Furthermore, there was a lot of wind during the experimentations, leading to an accentuation of the noise of the overall map. For this case, different configurations of risk were analyzed. Figure 5.28 shows the speed as well as the risk taken by the robot. In the first configuration, the robot entered the grass at $t \approx 12\,\mathrm{s}$. It was allowed to have an expected risk of $0.1\,\mathrm{kg\,m\,s^{-1}}$ and an upper risk of $5\,\mathrm{kg\,m\,s^{-1}}$. The grass had 5 % chance of having an infinite mass. The robot stopped at $t \approx 18\,\mathrm{s}$ as it was about to enter a denser zone, meaning a zone with a higher collision probability. All the high speed commands lead to a too high risk and the robot had to completely stop. During the traversal of the tall grass, the speed of the robot is maintained to a low value as the grass might hide an obstacle. As the robot goes out of the grass at $t \approx 36\,\text{s}$, it increased its speed to its maximum since a collision is more unlikely to happen. The same experiment was conducted where this time the grass zone had a probability of 99 % to have a null mass. The robot stopped in the same place as the first time, but increased its speed faster as it was surer that the collisions were harmless. By doing so, it reached the goal quicker. The third time, the robot was sure there was no obstacle in the grass. Hence, it crossed the environment at full speed since any collision was harmless even if the maximum risk allowed was null.

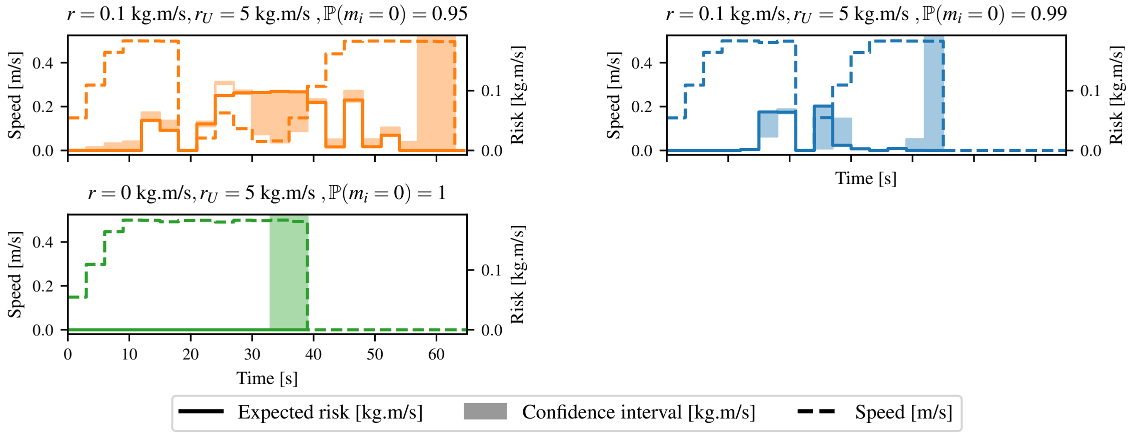


Figure 5.28: Speed (dashed line) and risk (solid line with shaded area for its confidence interval) of the chosen paths of the robot when it chooses to go through the tall grass, for different configurations. The more certain the robot is that there are no obstacles in the grass, the faster it reaches its goal.

A very specific event can appear as sometimes, the expected risk is outside the confidence interval. It can be seen for $t = 25\,\text{s}$ on the first graph of Figure 5.28. Computing the risk with lower lambdas can indeed lead to a higher risk in very specific conditions. In our case, the robot computed the risk for a path going out of the area where the mass of the obstacles was classified as low (i.e., the tall grass) by the camera. By doing so, any collision happening outside this zone will have a higher expected force of collision. It is then considered less risky to collide inside the area of low mass: lowering the lambdas leads to a higher chance to collide outside as the robot has a lower chance to be stopped inside the zone, leading to a higher risk. Appendix D provides a way to avoid such scenario and yield better confidence intervals.

5.5 Discussion

As mentioned before, the theory of the lambda-Field can be seen as a generalization of the framework of [267]. Under this consideration, it is possible to convert a Lambda-Field into a Bayesian occupancy grid and vice versa using Equation 5.52. In addition to generalizing the equations of [267], our framework allows the computation of expectation of a risk. It is possible as the Lambda-Field possesses a probability density function. Furthermore, the whole framework (i.e., mapping and path planning) perform in real-time at more than 10 Hz and a GPU implementation would speed up even more the algorithms.

The theory of Poisson Point Process has already been used by [109] for known obstacles. The Lambda-Field can be seen as the transfer of their work for occupancy grids.

However, one of the major drawbacks of the Lambda-Field is the assumption done in Equation 5.21, that is every cell in the error region of the range sensor carries the same information. Using such an approximation indeed leads to inflating the obstacles, meaning that some narrow corridors where the robot could go through become impracticable. The modeling of the sensor can also be discussed: for practical reasons, the sensor is assumed to have a deterministic error region where the collision is sure to have happened. Taking too small an error region would lead to augmenting the probability of wrong measurements, increasing the confidence interval of the lambdas hence decreasing the speed of the robot. Inversely, taking too big an error region leads the map to inflate the obstacles hence decreasing the space the robot can evolve in. However, in the case of lidar sensors, their precision is such that their error region often reduces to a low number or even a single cell, meaning that almost no inflation occurs. This can be seen on Figure 5.15 or Figure 5.18 for instances, where the structured obstacles does not appear bigger on the Lambda-Field than in the Bayesian occupancy grid. In the case where the range sensor has a bigger error zone, Appendix C gives a way to reduce the inflation by using a standard probabilistic sensor model.

The computation of the confidence intervals can also yield to deeper discussions. Indeed, an empty cell close to an obstacle can be considered to have greater chances to fail the reading and return a ‘hit’ measurement. Although this problem is already managed by the error zone, taking into account that cells close to the error zone have a greater chance to be misread can lead to more precise maps. This would lead to lower the upper bound of the lambdas in large empty zone, thereby increasing the traversal speed (i.e., efficiency) of the robot. As this work only dealt with constant false positives and negatives ratios, future works will investigate a deeper use of confidence intervals in harsher environments such as snowstorms where a lot of faulty 'hit' measurements coming from the snow can hinder the robot displacements.

So far, only the lidar measurements are used to estimate the lambdas and a camera for the mass estimation. However, in a more cooperative approach, the robot can also assess the hazardousness of the environment while navigating in it. For instance, if the robot went safely through a zone where the probability of causing a harmful collision p_{ti} is not null, the map can be updated accordingly and the probability of harmful collisions can drop. Note that adding such information needs to be done carefully, as informing that a zone is safe for a robot does not mean another robot with a different mass and mechanical configuration will be also safe to cross. Furthermore, as the framework creates maps centered on the robot, there is currently no tool for matching temporally distant observations (such as loop closures). In case where the construction of large scale maps is necessary, an external Simultaneous Localization And Mapping (SLAM) algorithm such as the Iterative Closest Point (ICP) can provide a corrected localization.

Also, some issues can appear while estimating the risk with our framework for global path planning algorithms. Indeed, it may be harder to understand the metrics. The longer the path, the higher the risk will be. As this behavior seems intuitive, it leads to several questions. For two paths with the same risk but a different length, are we willing to take the two paths with the same confidence? Or should we weight the risk by the length of the path, meaning that we are willing to take more risk for longer paths? We chose to understand the risk as the risk of a given command. Indeed, we believe that as humans, we assess the risk of every step we make without thinking about the length of the path.

Furthermore, the expectation of the risk is not fitted to model the 'long-tail' of the Gaussian, meaning the low-probability events that can happen. The faulty measurements of the lidar are handled with the confidence intervals of the lambdas but other metrics may better estimate the 'long-tail'. The Conditional Value at Risk presented by [130] explicitly measures the risk of the 'long-tail'. It can then be a better indicator of the risk when a low probability, high-risk situation arises, for example when a high mass obstacle hides in the grass.

5.6 Conclusion

In this chapter, we present a novel representation of the occupancy information of the environment, called Lambda-Field. We first derived a way to construct the map, as well as confidence intervals over these values. This representation allows the computation of expectation over a path, giving a natural way to assess different types of risks. The Lambda-Field is very alike the Bayesian occupancy grid for mapping, with the only notable difference that the Lambda-Field better stores unstructured obstacles like bushes, tall grass or wire fence. In addition, the Lambda-Field provides the computation of a generic risk that depends on the application.

In the case of unmanned ground vehicles, we chose to represent the risk as the force of collision. In contrast to risk metrics defined on Bayesian occupancy grids, our risk possesses a physical meaning. We were able to control the level of risk the robot could take over its planning, giving behaviors impossible with classical path planning environment representations. The robot was indeed allowed to cross low mass occupied areas such as tall grass as long as the risk level was low enough. Therefore, the Lambda-Field provides a framework that regulates the path as well as the speed of the robot, ensuring the robot's safety.

However, this framework assumes a static world: although this assumption is correct in numerous situations, such a technique would not work in urban environments. Therefore, we extend the Lambda-Fields to the dynamic world in the next chapter.

CHAPTER 6

DYNAMIC LAMBDA-FIELDS FOR THE NAVIGATION IN URBAN ENVIRONMENTS

6.1 Context of the work **155**
6.2 Proposed approach **158**
6.2.1 Computation of the expectation of a cell 160
6.2.2 Measuring particles 162
6.2.3 Particle Management 163
6.2.4 Risk assessment and Path Planning 165
6.3 Experimentations **167**
6.3.1 Evaluation 167
6.3.2 Validation of the approach 169
6.4 Discussion **171**
6.5 Conclusion **173**

6.1 Context of the work

In the previous chapter, we presented a framework that answers the limitations of the Bayesian Occupancy Filter (BOF) for risk assessment. Indeed, the Bayesian framework cannot infer physical risks because of the type of information it stores (i.e., a probability of collision) that does not allow for integration over a path. We developed a novel method that instead of storing a probability, stores an intensity that once integrated yields a risk that can be the probability of collision. However, our framework is for now only applicable in static world.

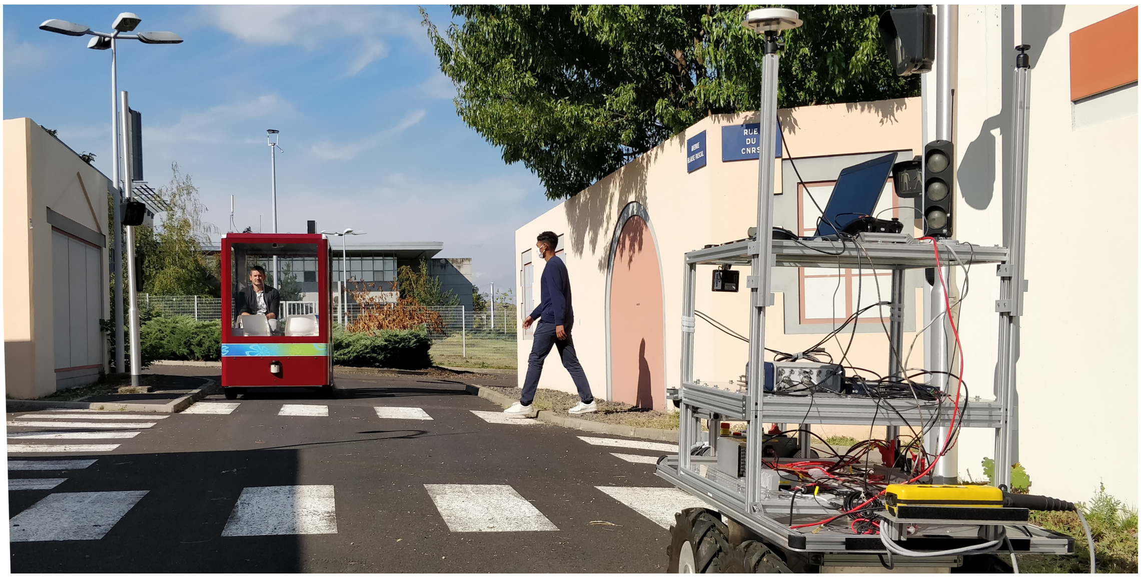


Figure 6.1: Example of an urban scenario that the robot can encounter. A pedestrian is crossing the road, while a vehicle is approaching the robot in the other lane. Using the Dynamic Lambda-Fields, the robot is able to estimate the risk of its actions, taking into account a generic risk that can be adapted depending on the scenario.

In this chapter, we extend the Lambda-Field to the dynamic realm, henceforth becoming Dynamic Lambda-Field. First, we present a use-case that justify the need of such a framework and why the BOF framework is not able to always yield consistent results. Then, the theory is given, extending the mathematics from the static case to dynamic environments. Experimentations validate the approach with convergence tests, and real worlds experiments demonstrate the usefulness of the framework. Then, we conclude this chapter with numerous avenues for improvement. Indeed, the framework is heavily generalizable in both the static and dynamic cases as discussed in the general conclusion of this part.

In the current state of the robotics, one could say that *most* of the application are solved. Indeed, autonomous vehicles actually roam the highways and start to be able to cope with complex crossroads. What still needs to be solved is what one can characterize as the long-tail of the events, namely the high-risk, improbable scenarios. Our framework places itself right into this niche. Indeed, in the case where the robot can travel safely, the risk is by definition zero. There is no difference between a physical risk from the Lambda-Fields or for instance the computation of the risk from [274]. A zero risk is zero no matter the underlying theory. As such, our framework only shines in hazardous scenarios that are more complex to handle.

Figure 6.1 depicts an example of hazardous scenarios in the context of urban navigation. The robot, on the right of the figure, wishes to cross the crossroad. At the same time, another vehicle is approaching the robot on the other lane. Then, a pedestrian

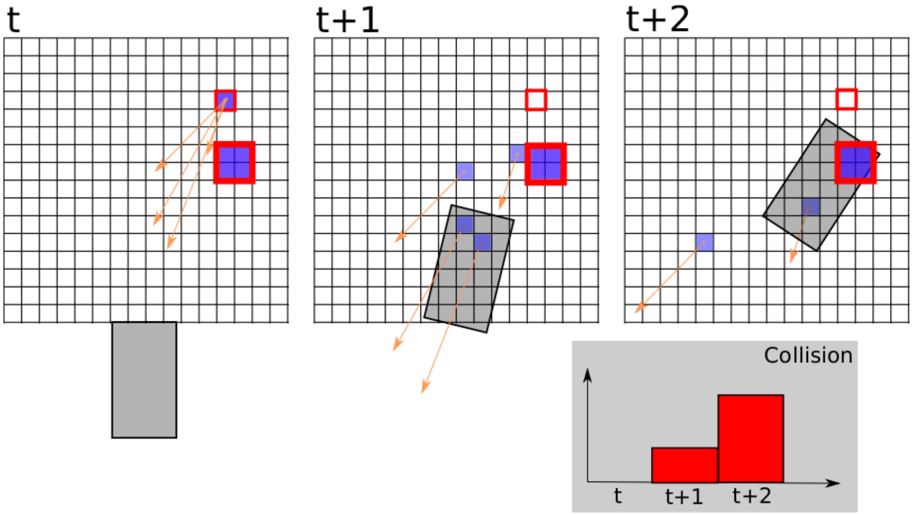


Figure 6.2: Example of collision probability prediction for a vehicle in light gray, from [281]. Using the first metric, they compute the probability of colliding with a specific area for each time step.

emerges and crosses the road. At this time, the robot has too high a speed to stop before colliding with the pedestrian. Thus, the robot has two options:

- brake as much as possible and collide with the pedestrian; and
- brake and swerve, colliding with the vehicle in the other lane.

As already discussed in Section 2.3, no solution is better than the other. In the one hand, an utilitarian approach would swerve and collide with the other car, as in overall the risk of injury is lower for the occupants of the vehicles. On the other hand, a 'self-safety' approach would collide with the pedestrian as the damages to the robot would be negligible. As such, a risk assessment framework should be able to infer both risks and leave the choice of which one to use to the philosophers or the insurance companies.

In the context of Bayesian Occupancy Grids, Rummelhard *et al.* [281] proposed a method to assess probabilities of collision in dynamic environments. In order to tackle the problem of the Bayesian tessellation, they proposed two approaches that are:

Collision probability with a given surface: In the same fashion as Lachapelle *et al.* [274] that pondered the probability of collision by the areas of the cells, they proposed to evaluate the likelihood of colliding with a selected surface size. According to this area, and to the area of a cell, the number of cells n to be struck is thus given, the computed risk value being the probability to collide with n cells of the grid. Figure 6.2 provides an example of such metric.

Maximum of collision probability: They argue that in real experimentations, a simpler approach that is the maximum value of collision over the cells, tends to work better.

Following, Guardini *et al.* [105] proposed to convert a Bayesian occupancy grid into a generic risk map where the risk function can be changed according to the scenario, in the same fashion as our work. For each cell, they define the associated Probability of Collision with Injury Risk (PCIR) as

$$\text{PCIR} = \mathbb{P}(\text{occ}) \cdot S(c_i), \tag{6.1}$$

where $S(\cdot)$ is a severity function that characterizes the collision at the given cell c_i. This severity function has the same meaning that our risk function, that is quantifying the risk of the event collision. However, in the case of risk maps deduced from Bayesian occupancy grids, the issue remains the same as before: there is no natural way to infer from this map a consistent risk for a given path. One can of course extend the work of Rummelhard *et al.* [281] and infer the risk of the path as the maximum risk of the cells encountered. However, drawbacks emerge quite fast as, for instance, colliding with a pedestrian with a severity of 95 % is judged safer than colliding $N > 1$ pedestrians with a severity of 94 %.

Subsequently, we propose in the next section what can be seen as the natural extension of the work of Guardini *et al.* [105]. We present a way to infer generic, consistent risks over a path in dynamic environments. Instead of storing a risk for each position, we instead store the intensity of the event collision and convert it to a risk[1] only at the time of the assessment of a given path. Thus, consistent metrics are returned that keep their physical meaning and yield the robot to make more informed decisions.

6.2 Proposed approach

We present here how to construct Dynamic Lambda-Fields as well as how to assess risks using this map as done in the static case in Chapter 5. In this chapter, we define the risk as the maximum change of kinetic energy of the robot and the obstacle (static or dynamic) due to a collision. The framework allows the risk function to be changed depending on the application. For example, the risk function could also model the probability of survival for both the vehicle occupants and the collided obstacle if it is human. By keeping the risk in its physical form, the robot is able to make more informed decisions without the need of tuned user-defined thresholds. First, we show how to compute the Lambda-Field, namely the intensities λ_i of both static and dynamic obstacles, using a lidar sensor. Then, we present how to manage the particles used to represent the obstacles in the environment. Finally, we show how to use this field to assess physical risks given a generic risk function, leading to the generation of safe paths for the robot.

---

[1] This risk can be reduced to a probability of collision, and therefore the framework is also a extension of Rummelhard *et al.* [281].

As the environment contains both static and dynamic obstacles, the value of the lambda of a cell c_i (i.e., its intensity representing the likelihood of a collision occurring in this cell) is stochastic. Indeed, the cell can be occupied by a static obstacle, a dynamic obstacle or be free of obstacles. Therefore, we define the probability of collision as the probability of colliding with an obstacle in the expectation of the intensity field, leading to

$$\mathbb{P}(\texttt{coll}) = 1 - \exp\left(- \sum_{c_i \in \mathscr{C}} \Delta a \mathbb{E}\left[\lambda_{c_i}^{t_i}\right] \right), \tag{6.2}$$

for a path crossing the cells $\mathscr{C} = \left\{c_0, \dots, c_{N-1}\right\}$ at the times $\left\{t_0, \dots, t_{N-1}\right\}$, of expected intensities $\left\{\mathbb{E}\left[\lambda_{c_0}^{t_0}\right], \dots, \mathbb{E}\left[\lambda_{c_{N-1}}^{t_{N-1}}\right]\right\}$ and of area Δa. One can note that the probability of collision does not depend on the tessellation size as the sum is weighted by the area of the cells, as shown in [129].

In order to estimate the expectation of the intensity $\mathbb{E}\left[\lambda_c^t\right]$ for each cell c, we use particles that represent the possible obstacles in the environment. The static grid is considered as a set of particles that do not move and have the size of a cell, whereas dynamic obstacles are defined as moving particles of different classes. We define in this work three different classes of particles: the ‘cell’ class, for the static environment, the ‘pedestrian’ class and the ‘car’ class for the dynamic obstacles. More classes can be added at the cost of a slower convergence of the obstacle classes. Each particle has a position, a speed, a velocity profile (i.e., a car can achieve greater speed than a pedestrian, whereas a pedestrian can change direction quicker). Each particle is also defined by their size. For instance, a pedestrian particle is represented by a 40 cm diameter circle whereas a vehicle is described by a rectangle of 2 × 1 m. Note that the particles’ footprints are not dependent on the tessellation size. Naturally, particles of the ‘cell’ class have null velocity and acceleration. Using these attributes, the framework is able to represent the different obstacles more efficiently, as only one particle can represent an obstacle even if it spans several cells, contrary to [189] where a particle does not have a size. Also, using the particle classes representing the obstacle, the classes of the obstacles are inferred at the same time, allowing the framework to take into account their classes while assessing the risk of a path. For instance, if the risk function takes into account the probability of survival of collided obstacles, it allows the robot to prefer colliding with a car over a pedestrian. Note that no clustering is performed and that the class inference is solely done using lidar detections.

As an example, Figure 6.3 gives the Lambda-Field of Figure 6.1 where the robot wants to cross the environment while a pedestrian abruptly appears on the road, therefore having only two choices: either braking and colliding with the pedestrian or swerving and colliding with the car in the other lane, hence saving the pedestrian. The environment is populated by static particles (one per cell) of different intensities whereas the possible dynamic obstacles are represented using dynamic particles (in this example, only two, one of type 'pedestrian' and the other of type 'car'). The expectation of the cells' intensity that the robot crosses is derived in the following section, taking into account the intensity of the particles that are in the cell at the time of traversal (a static particle and possibly several dynamic ones). Once the expected intensity of the cells is computed, the probability of collision of the robot's path is given by Equation 6.2, integrating the expected intensity of the cells crossed by the robot. Then, the expectation of a risk function over the path can be computed, giving more insight on which path is the safest.

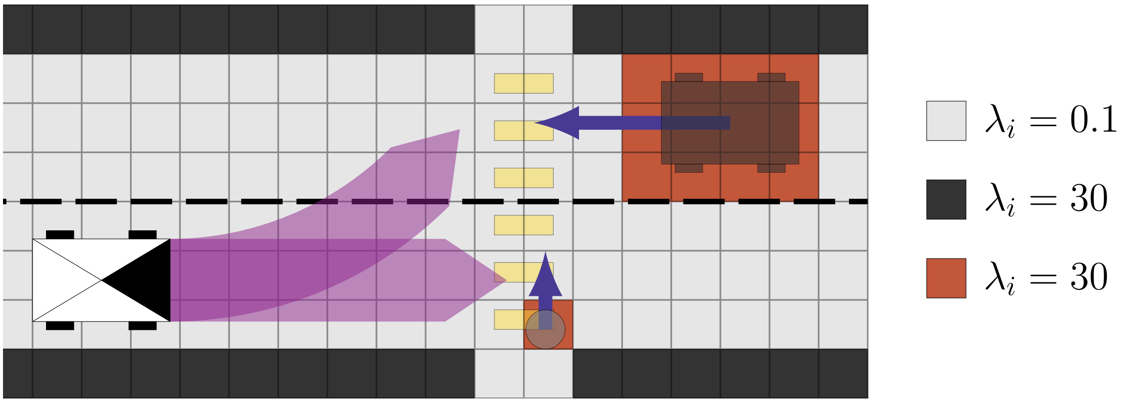


Figure 6.3: Example of Dynamic Lambda-Field where a pedestrian emerges unexpectedly on the road in front of the robot (black box with its front represented as a filled triangle) while another vehicle is approaching in the opposite lane. The environment is represented as a set of cells storing the expected intensity, computed using the intensity of the static (resp. dynamic) particles, depicted in gray (resp. red) scale, occupying these cells. Depending on the risk function, the robot can assess which one of the two paths (in purple) is the safest.

6.2.1 Computation of the expectation of a cell

In order to measure particles and infer the risk of a path, the first step is to compute the expectation of the intensities of the cells. When measuring the environment at a given time, we assume that a cell cannot contain more than one obstacle (e.g., a cell can contain a wall or a car but not both). This consideration allows us to only measure the obstacle (i.e., the particle) that is in the cell, as for instance we do not want to measure the static part of a cell as occupied if a moving car is in the same cell at this time. Furthermore, we also model the probability that a particle still exists: it is indeed unlikely that a particle still represents an obstacle after having run into a wall. If this case occurs, the existence probability of the particle will drop. Under these considerations, the expected lambda of a cell c is given by

$$\mathbb{E}\left[\lambda_c^t\right] = \sum_{p_i \in c} \lambda_i \mathbb{P}\left(p_i \in c\right) \mathbb{P}\left(e_i \,\middle|\, p_i \in c\right), \tag{6.3}$$

where λ_i is the lambda of the particle p_i being in the cell at the time of traversal t, $\mathbb{P}\left(p_i \in c\right)$ depicts the probability that the particle p_i is the one located in the cell and $\mathbb{P}\left(e_i \,\middle|\, p_i \in c\right)$ is the probability that the particle p_i exists. In the case where there is no dynamic obstacle in the cell, the probability $\mathbb{P}\left(p_i \in c\right)$ will be one for the static particle, which can have a very low lambda, thereby meaning that the cell is safe to cross, or a high lambda, meaning that the cell contains a static obstacle. Assuming that every cell has exactly one obstacle, we can compute the probability of the particle p_i of being in the cell c as the probability that only the particle p_i creates a collision given that there is exactly one obstacle in the cell:

$$\begin{aligned}
\mathbb{P}\left(p_i \in c \,\middle|\, (p_0 \in c) \oplus \cdots \oplus (p_{N_P-1} \in c)\right) &= \frac{\left(1 - \exp\left(-\lambda_i \Delta a\right)\right) \prod_{j \neq i} \exp\left(-\lambda_j \Delta a\right)}{\sum_k \left(1 - \exp\left(-\lambda_k \Delta a\right)\right) \prod_{j \neq k} \exp\left(-\lambda_j \Delta a\right)} \\
&= \frac{\left(1 - \exp\left(-\lambda_i \Delta a\right)\right) \exp\left(\lambda_i \Delta a\right) \prod \exp\left(-\lambda_j \Delta a\right)}{\sum_k \left(1 - \exp\left(-\lambda_k \Delta a\right)\right) \exp\left(\lambda_k \Delta a\right) \prod \exp\left(-\lambda_j \Delta a\right)} \\
&= \frac{\left(1 - \exp\left(-\lambda_i \Delta a\right)\right) \exp\left(\lambda_i \Delta a\right)}{\sum_k \left(1 - \exp\left(-\lambda_k \Delta a\right)\right) \exp\left(\lambda_k \Delta a\right)} \\
&= \frac{\left(1 - \exp\left(-\lambda_i \Delta a\right)\right)}{\exp\left(-\lambda_i \Delta a\right)} \cdot \frac{1}{\sum_k \left(1 - \exp\left(-\lambda_k \Delta a\right)\right) \exp\left(\lambda_k \Delta a\right)} \\
&\propto \frac{1 - \exp\left(-\lambda_i \Delta a\right)}{\exp\left(-\lambda_i \Delta a\right)},
\end{aligned} \tag{6.4}$$

for a cell c of area Δa where N_P particles lie in it, $\oplus$ being the standard XOR operator. The probability of existence $\mathbb{P}\left(e_i \,\middle|\, p_i \in c\right)$ is defined as the joint probability that the particle did not collide with the static environment since its creation and that it still follows an obstacle, computed as

$$\mathbb{P}\left(e_i \,\middle|\, p_i \in c\right) = \exp\left(-\sum_{c \in \mathscr{P}_i} \lambda_{c,s}\right) \cdot \exp\left(-\tau \cdot t_i\right), \tag{6.5}$$

where $\mathscr{P}_i$ is the path (set of crossed cells) of the particle p_i, $\lambda_{c,s}$ is the lambda of the static particle at the cell c (we allow dynamic particles to cross without collision), t_i is the time since the last measurement ‘hit’ (i.e., the lidar hits the obstacle) of p_i and

τ is the rate of the distribution. Indeed, if the particle has not been measured for a long time, it is very likely that either the particle lost the obstacle or that the obstacle left the field of view of the robot. This probability can also take into account other sensors such as a camera: for instance, the probability of existence of particles of type 'pedestrian' should drop if the camera informs the robot that there is a car at this position. Finally, the probability of existence can also be updated with methods such as [282] that performs an ICP algorithm while segmenting lidar points as static or dynamic. Therefore, if a point is segmented as dynamic, the probability that the static particle is the one lying in the cell falls to zero, and inversely if the point is read as static.

6.2.2 Measuring particles

Using the expressions of the probability of collision given by Equation 6.2 and the expectation of the intensities of the cells given by Equation 6.3, we provide a means of estimating the lambdas of the static field as well as the dynamic particles using a lidar sensor. As in [129], we determine the combination of the intensities $\lambda = \left\{\lambda_i\right\}$ of the particles that maximizes the expectation of the K beams the lidar has shot since the beginning. The lidar is modelled as a range sensor with an error region of area e: for a lidar measurement, the true position of the obstacle is contained within the error region, itself centered on the measurement. In the case of a perfect sensor, this region is reduced to a point at the measurement position. Using the derivation in [129] and the expectation of the intensity given by Equation 6.3, the intensities of the particles are given by

$$\lambda_i = \frac{1}{e} \ln\left(1 + \frac{h_i}{m_i}\right), \tag{6.6}$$

where h_i is the sum of probabilities $\mathbb{P}\left(p_i \in c_h\right)\mathbb{P}\left(e_i \,\middle|\, p_i \in c_h\right)$ each time the particle p_i has been counted as 'hit' in the cell c_h (i.e., was in the region of error of the sensor) and m_i the sum of probabilities $\mathbb{P}\left(p_i \in c_m\right)\mathbb{P}\left(e_i \,\middle|\, p_i \in c_m\right)$ each time the particle p_i has been counted as 'miss' in the cell c_m (i.e., the lidar beam crossed the cell without collision). For this equation to be true, one needs the assumption that for a given measurement, all of the particles measured in the error region of the sensor have the same intensity. As the lidar error region tends to be small, this assumption holds in every situation except for low-intensity static particles. Indeed, high-intensity dynamic particles can easily come to the cell containing the low-intensity particle, thereby the assumption that every obstacle has the same intensity no longer holds. This case is tackled by assuming that the low-intensity static particle has the same intensity as the high-intensity particles, therefore overestimating its intensity and keeping a conservative approach.

6.2.3 Particle Management

Using the particles update equation previously derived, the Lambda-Field is updated as described below. At each iteration, the particles evolve, are updated with lidar measurements and are resampled. First, the particles evolve using a simple update equation:

$$\begin{aligned} &\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t \Delta t, \\ &\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + \mathbf{a}_i \Delta t \quad \text{with } \mathbf{a} \sim \mathcal{N}(0, \mathbf{\Sigma}_i), \end{aligned} \tag{6.7}$$

for the particle p_i of position and speed $\mathbf{x}_i^t, \mathbf{v}_i^t \in \mathbb{R}^2$ at the time t, where $\mathbf{a}_i \in \mathbb{R}^2$ is a centered Gaussian random variable of covariance $\mathbf{\Sigma_i}$ depicting the acceleration of the particle. Evidently, static particles have zero speed and acceleration. Then, for each measured cell c, we compute the probability that the particle p_i is indeed the particle in the cell. Each particle in the 'hit' zone (resp. 'miss' zone) of the lidar measurements increments its 'hit' counter h_i (resp. 'miss' counter m_i) of the quantity $\mathbb{P}\left(p_i \in c\right)\mathbb{P}\left(e_i \,|\, p_i \in c\right)$ as shown in Equation 6.6. Moreover, particles have a low probability to switch classes (i.e., 'pedestrian' switching to 'car' and vice versa). This consideration avoids an incorrect convergence of the particles (e.g., a low velocity car can be represented as a group of pedestrians, however both hypotheses have to be maintained).

Once the particles have been updated, they are resampled according to the joint probability of their existence and selecting one of the cells in which they are located:

$$w_i \propto \mathbb{P}\left(p_i \in c\right)\mathbb{P}\left(e_i \,|\, p_i \in c\right) \sum_{p_i \in c} 1 - \exp\left(-\Delta a \mathbb{E}\left[\lambda_c^t\right]\right) \tag{6.8}$$

Moreover, particles can be born during the resampling step. Indeed, obstacles can appear on the map and the particles might not converge to an obstacle, leaving a dynamic obstacle unidentified. Even if this situation is less likely the more particles we have, dealing with such a critical case is necessary. Hence, we allow the birth of particles inside any cell c measured 'hit' by the lidar with

$$w_c^{\texttt{birth}} = \gamma \cdot \exp\left(-\Delta a \sum_{c \in \mathscr{E}_k} \mathbb{E}\left[\lambda_c^t\right]\right), \tag{6.9}$$

for every 'hit' measurement of error region $\mathscr{E}_k$, where γ is a coefficient controlling the proportion of births at each resampling. If the measured cell is already populated by particles of high lambdas, meaning that the obstacle is already represented, the birth probability drops to zero. If the particle which is born is picked during the resampling, random class and speed are drawn from a uniform distribution. algorithm 1 summarizes the procedure.

Algorithm 1: Particles management

```
repeat
    Evolve particles using Equation 6.7
    Update particles with measurements using Equation 6.6
    forall particles $p_i$ do
        Draw a particle $p_k$ with weight $w_k$ and birth weight $w_c^{\texttt{birth}}$
        Replace particle $p_i$ with particle $p_k$
    end
until True;
```

Once every particle has been updated and resampled, we infer the underlying distribution for each cell. Indeed, planning while taking into account every particle would not reach real-time constraints. For each cell, the speed is modeled as a normal distribution $\mathcal{N}(\mu_v, \sigma_v)$ whereas the direction is modeled using the Von-Mises distribution (as the direction lies on a circle) of parameters $\mu_\theta, \kappa_\theta$, where these parameters are estimated as in [279]. We estimate the above parameters as

$$\begin{aligned} \mu_v &= \frac{1}{N}\sum v_i, & \sigma_v^2 &= \frac{1}{N-1}\sum (v_i - \mu_v)^2, \\ \mu_\theta &= \arg\Big(\sum \exp(j\theta_i)\Big), & \kappa_\theta &= \frac{\bar{R}(2-\bar{R}^2)}{1-\bar{R}^2}, \end{aligned} \tag{6.10}$$

where v_i, θ_i is the speed and direction of the particle p_i and $\bar{R} = \left|\frac{1}{N}\sum_k \exp(j\theta_k)\right|$, j being the standard imaginary number solving the equation $j^2 + 1 = 0$. In the case where κ_θ is large enough, the Von-Mises distribution can be approximated by a normal distribution of the same mean with a standard deviation of $\sigma_\theta = \sqrt{1/\kappa_\theta}$. Using Von-Mises distribution avoids periodicity problems while dealing with high uncertainty angles.

At the end of the mapping process, a grid is created where each cell contains an intensity for the static part, as well as an intensity for each type of dynamic particle (i.e., summing the intensities of the particles of the same class, using Equation 6.3). We do not sum the intensities of the different classes since the risk can depend on the type of obstacle. We instead provide each class intensity to the planner. As we defined three classes in this work (static, pedestrian and car), each cell contains three intensities, one for each obstacle.

Figure 6.4 provides a graphic example of the overall algorithm. In this scenario, a car traveling at $1\,\mathrm{m\,s^{-1}}$ south, is measured by the robot. Particles are evolved and updated as shown above, as shown in black in the figure. In this example, only car particles have been created. Then, for each cell the underlying distribution of the class 'car' is inferred, where one of them is depicted below the robot. As shown in the next section, we transform the distribution in a set by taking two sigmas (i.e., approximatively 95 %) on the speed and orientation, yielding the set depicted in red in the polar plot. Because of the occlusion due to the vehicle, the robot is unable to measure particles behind the robot. These particles represent the possibility that another car is hidden behind the other. Depending on the rate τ of the distribution that dictates the existence of non-measured particles as shown in Equation 6.5, these hypothesis can either be kept or removed. In this example, the rate was set to $\tau = 0$ so that all hypothesis are kept even if there are not measured since a long time. In real-world applications, a rate $\tau > 0$ has to be set to avoid particles living and being re-sampled in far away regions that hinder the convergence of true obstacles near the robot (since there is far less particles that birth on these obstacles).

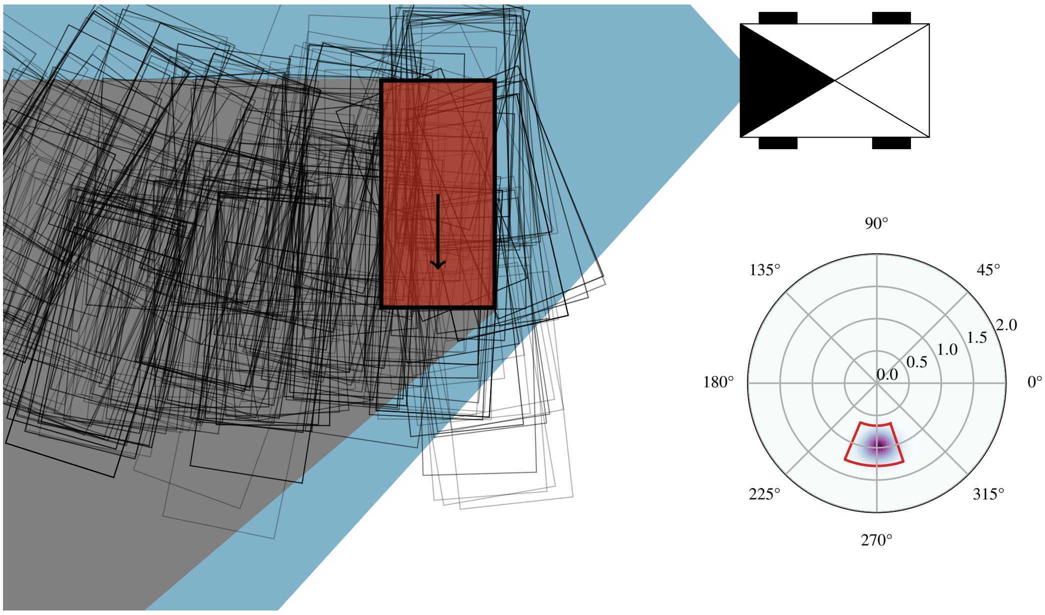


Figure 6.4: Example of velocity estimation. A car (red box) is navigating at a speed of 1 m s^{-1} in the south direction. Because of the occlusion (in gray) caused by the vehicle, all the hypothesis of other vehicles behind that said car are kept, while the speed and direction of the vehicle is estimated. The red line in the polar plot corresponds to set at 95 % on the speed and orientation distribution. 500 particles were used in this toy example.

6.2.4 Risk assessment and Path Planning

Using the data provided by the dynamic map, we generate safe trajectories for the robot. As in Gerkey *et al.* [280], we sample commands of translational and rotational velocities (v, w) and choose the one leading the closest possible to the objective while being below the allowed risk. In each cell, the robot has to go through all of the obstacles simultaneously (i.e., static obstacle and all of the dynamic ones arriving in the cell at the same time as the vehicle). The probability of colliding with the obstacle o_k in a cell c containing N_O obstacles $o_0, \dots, o_{N_O-1}$ of intensities $\lambda_0, \dots, \lambda_{N_O-1}$ at the time t is the joint probability of not colliding the other obstacles $\{o_i\}_{i \neq k}$ and colliding with the obstacle o_k:

$$\begin{aligned}\mathbb{P}\left(\texttt{coll}_{c,k}\right) &= \int_0^{\Delta a} \exp\left(-a \sum_{i \neq k} \lambda_i\right) \cdot \lambda_k \exp\left(-a\lambda_k\right) \mathrm{d}a \\ &= \frac{\lambda_k}{\mathbb{E}\left[\lambda_c^t\right]} \left[1 - \exp\left(-\Delta a \mathbb{E}\left[\lambda_c^t\right]\right)\right], \qquad (6.11)\end{aligned}$$

where $\mathbb{E}\left[\lambda_c^t\right] = \sum_{i=0}^{N_O-1} \lambda_i$ is the expected lambda of the cell c at the time of traversal t, where the intensities λ_i of the incoming obstacles o_i have been pondered by the probability of reaching the cell beforehand, using the probability density function of the Lambda-Field defined in Equation 5.30. One can note that the expectation is the same as Equation 6.3 where we lifted the assumption that only one obstacle can be in the cell. Indeed, even if only one obstacle can truly be in a cell, several obstacles can reach the cell from different directions and create a collision with the robot which attempts to cross it. Since several obstacles can hit the robot in the same cell, the assumption of each cell having only one obstacle is lifted, yielding $\mathbb{P}\left(p_i \in c\right) = 1$ for the computation of expectation in risk assessment. Using Equation 6.11, we compute the risk of a path as the expectation of the risk function $r(a, o_k)$, where a is the traversed area at the time of the collision and o_k is the obstacle the collision occurs with, as

$$\begin{aligned}&\mathbb{E}\left[r(\cdot)\right] = \sum_{i=0}^{N-1} K_i \sum_{o_k \in c_i} \frac{\lambda_k}{\mathbb{E}\left[\lambda_{c_i}^{t_i}\right]} \cdot r(i\Delta a, o_k), \\ &\text{with } K_i = \exp\left(-\sum_{j=0}^{i-1} \Delta a \mathbb{E}\left[\lambda_{c_j}^{t_j}\right]\right)\left(1 - \exp\left(-\Delta a \mathbb{E}\left[\lambda_{c_i}^{t_i}\right]\right)\right),\end{aligned} \qquad (6.12)$$

for a path passing through the cells $\{c_0, \ldots, c_{N-1}\}$ at the times $\{t_0, \ldots, t_{N-1}\}$ of expected lambdas $\mathbb{E}\left[\lambda_{c_i}^{t_i}\right]$. The curvilinear abscissa s is linked to the traversed area by the relation $s = i\Delta a / L$ where L is the width of the robot. In order to determine which obstacles are in the cell at the time of traversal, we convert the velocity distribution of each dynamic cell into a set as depicted in Figure 6.5. The set corresponds to the shape of the obstacle's (i.e., dynamic cell) path using two sigmas on its velocity and orientation. This shape is then tested to cross the cells of the robot's path, using the Gilbert-Johnson-Keerthi distance algorithm known for its fast computation. In the case of a collision between the robot's path and the cell's path (dashed cells in Figure 6.5), the earliest arrival and latest departure time of the dynamic obstacle and the robot are compared. If the time intervals intersect, the obstacle is said to be in the cell of the robot's path. Using this risk assessment method, the robot is able to effectively assess the risk of a path. In the following section, the risk is defined as the change of kinetic energy arising from the collision, thus taking into account the harmfulness of the collision for both the robot and the obstacle it collides with.

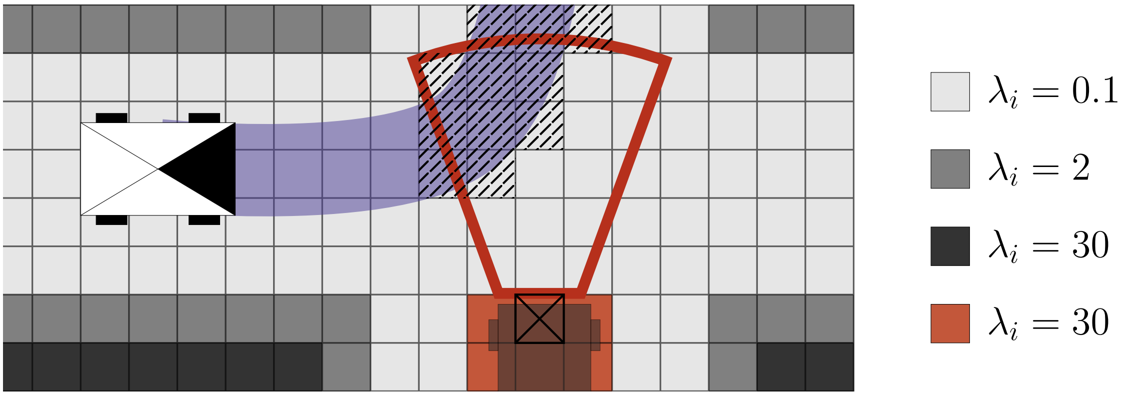


Figure 6.5: Example of risk assessment. The robot (left vehicle) wishes to cross the crossroad using the trajectory depicted in blue. A dynamic obstacle arrives from the bottom of the map, with a probabilistic direction and speed, converted into a set (in red) that corresponds to the possible positions of the obstacle. Using these distributions for each dynamic cell (here depicted for the crossed one), the potential collision positions are marked (dashed cells). If the times of traversal intersect, the obstacles are added to the crossed cells for risk assessment.

6.3 Experimentations

In order to show the applicability of our theory, we implemented our mapping framework on a Jetson TX2 GPU. We used the robot depicted in Figure 6.1 (right), equipped with a lidar LMS-151 located at its front. As the aim is to plan short distance trajectories and the map is centered on the robot, we only used the odometry for the relative displacements of the robot. To estimate the states of the dynamic obstacles, we used 2×10^4 particles in the experiments, running at more than 10 Hz and more than 5 Hz if the planning component is carried out on the same GPU. The map size was set to 200 × 200 cells of size 15 × 15 cm, resulting in a map of 30 × 30 m. In order to accelerate the convergence of the particles towards the obstacle, particle velocities were resampled with a Gaussian noise of $\sigma = 0.3\,\mathrm{m\,s^{-1}}$.

6.3.1 Evaluation

First, we show that the convergence speed of the framework. Using the same type of validation as Nuss *et al.* [194], we simulated an environment where an obstacle was approaching the robot at a known velocity and orientation. This experiment was repeated 50 times in which the obstacle was either a pedestrian or a car. The velocity of the obstacle was chosen to be $1.5\,\mathrm{m\,s^{-1}}$, as this profile of speed can be matched by either the ‘pedestrian’ or the ‘car’ class. Using the fact that only one dynamic obstacle is in the environment, we retrieve the mean and a confidence interval at two sigmas for the velocity and the orientation. Figure 6.6 shows the results for a pedestrian (in green) and a car (in blue). We can see that both the speed and the orientation converge to a valid value after less than 2 s. Furthermore, as the framework assesses the risk by allowing two sigmas on the velocity and the orientation, the ground truth is always contained in the estimation. At first, the speed of the obstacle is overestimated as both cars of high velocities and pedestrian of low velocities coexist. In the case of the pedestrian, all particles of class ‘car’ are discarded as their size is too large for the obstacle (i.e., they lie in cells counted as free by the sensor). Once the ‘car’ particles are removed at $t \approx 1$ s, the mean velocity converges to the true value at the next iteration. In the case of the car, the car particles of high velocities are discarded as soon as they leave the measurements zone, which is indicated by the fact that the convergence of the mean is smoother than in the pedestrian experiment.

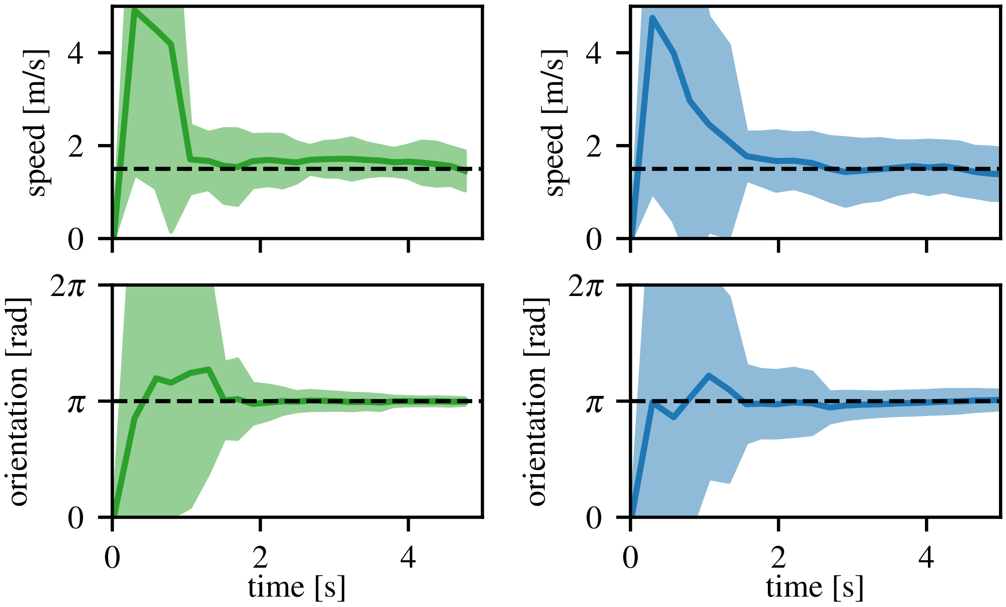


Figure 6.6: Convergence of the speed and orientation of a single dynamic obstacle (Left: pedestrian; Right: car) where the ground truth is depicted in dashed black. The mean is displayed in solid line whereas the confidence interval at two sigmas is depicted in light shade. The experiment has been repeated 50 times for each figure.

The convergence of the obstacle class is also studied. We compute the probability of the obstacle to be of a certain class as the probability to collide with dynamic cells of the class. Figure 6.7 shows the resulting convergence for a pedestrian and a car. In the case of the pedestrian, the framework quickly converges to the real class of the obstacle after 1 s. However, in the case of the car, the framework cannot decide whether the obstacle is a car or a group of pedestrians. Indeed, with only a lidar, the obstacle can match both classes equivalently. Other sensors such as a camera could remove the ambiguity. Also, the class would be determined to be a car if the velocity profile only matched the ‘car’ class (i.e., the obstacle moves at greater velocity).

Therefore, the framework is able to effectively infer the velocity, orientation and class (when possible) of the different obstacles.

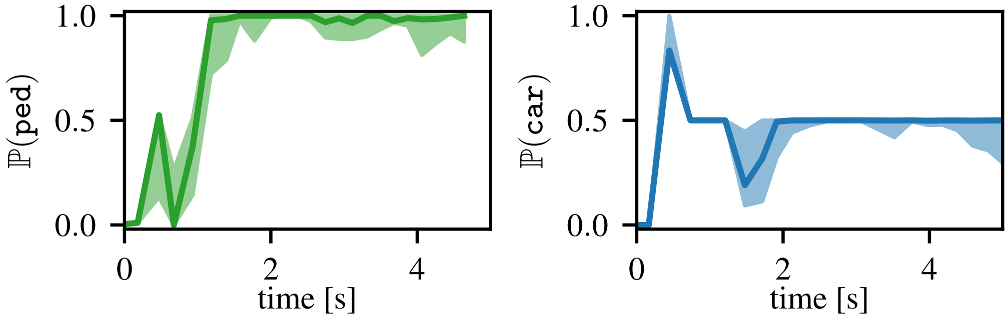


Figure 6.7: Convergence of the class probability for a pedestrian (left) and a car (right) with the quantiles at 25% and 75% in light shade. Whereas the pedestrian has its class quickly inferred, the framework cannot decide between a car and a group of pedestrians in the case where the car has a speed matching both classes.

6.3.2 Validation of the approach

In the following experiment, the risk is defined as the maximum gain of kinetic energy between the robot and the obstacle it collided with, assuming inelastic collisions, as

$$r(s, o_k) = \max\left(\frac{1}{2} m_R \left(v_R(s) - v_f\right)^2, \frac{1}{2} m_k \left(v_k - v_f\right)^2\right)$$
$$\text{with } v_f = \frac{m_R v_R(s) + m_k v_k}{m_R + m_k} \tag{6.13}$$

where $m_R, v_R(s)$ is the mass and velocity of the robot at the curvilinear abscissa s, m_k, v_k the mass and velocity of the obstacle o_k, and v_f the final velocity of the obstacles after collision. The mass of the robot was set to 150 kg while the masses of the 'pedestrian' and 'car' classes were set to 80 kg and 500 kg respectively. This risk enables the consideration of both the possible damages suffered by the robot but also by the obstacle it collided with. The decision to collide with a pedestrian takes into account that although the robot will suffer little damage, the pedestrian is at a much greater risk. Note that the risk function can be adapted depending on the context and can take into account other elements if available such as slippage, car deformation and so on. The static environment is assumed to have infinite mass, meaning that collisions with the static environment will always lead the vehicle to stop. As shown in Figure 6.1, the robot had to move through an urban-like environment consisting of a crossroad, where other agents such as pedestrians and cars were also evolving. A car was approaching in the other lane and a pedestrian entered the field of view of the robot from behind, afterwards crossing the road in front of it. For obvious security reasons, in this experiment, the velocity of the robot was bounded such that it will always be able to instantly stop in the case of hazardous situations where every path is too risky, contrary to the scenario depicted in Figure 6.3.

Figure 6.8 shows the resulting Lambda-Field for two different timestamps. The static environment is depicted with a gray scale, whereas the dynamic environment is shown with a red scale. Using our framework, the velocity distribution of each cell is extracted as well as a lambda of the static environment and each type of particle (i.e., in our case a lambda for the 'car' class and a lambda for the 'pedestrian' class). At $t = 19\,\mathrm{s}$, the large dynamic high-lambda zone (in red) of the map corresponds to the car, where the probability of being a car is approximatively 50 % for the underlying cells. The probability did not converge to 100 % because without additional sensors, the framework cannot decide whether the obstacle is a car or a group of pedestrians, as the velocity of $1.2\,\mathrm{m\,s^{-1}}$ is possible for both classes. At $t = 28\,\mathrm{s}$, a pedestrian emerges from behind the robot, then crosses the road right in front of it a few seconds later. When the robot detects the new obstacle, an inconsistency between the map and the measurements is found, leading the framework to create many particles on this location. Note the light gray trace which is left behind the pedestrian, as the framework had not yet decided whether the obstacle was static or dynamic. As shown in the right polar distribution (i.e., angle for the orientation, radius for the velocity), the hypothesis of the pedestrian pursuing its northward trajectory (top of the figure) is maintained. This hypothesis models the fact that without other sensors, the obstacle can in fact represent two pedestrians walking together, with the probability that they can change directions. The wrong hypothesis (i.e., pedestrian moving north) is discarded over the next iterations.

Finally, we used this dynamic map to plan safe paths for the robot. The goal of the robot was set at the top of the map, 15 m away from its position. To do so, we set the maximum risk to be $r_{\text{max}} = 1\,\mathrm{J}$ where any path below this risk is considered safe. Figure 6.9 shows the risks the robot underwent during the traversal. First, no obstacles were in sight, leading the robot to accelerate to its maximum speed, here at $0.5\,\mathrm{m\,s^{-1}}$. At $t = 10\,\mathrm{s}$, a car entered the field of view of the robot. As the obstacle was far away enough during the convergence of the speed and orientation, the robot did not stop its course. At $t = 22\,\mathrm{s}$, the car passed on the left side of the robot. As the velocities were precise enough not to encounter the path of the robot, it continued its course at full speed. At $t = 31\,\mathrm{s}$, the pedestrian took a hard left turn, deviating from its expected trajectories thereby leading the framework to birth particles on its position. Consequently, the robot detected a danger on every path it could take, leading it to stop as this decision is the one minimizing the risk. The associated risk is then the risk of the pedestrian running into the robot, thus harming himself. In contrast, the Bayesian occupancy grid would only yield a probability of collision (in this case equal to one) and the robot could not distinguish between the collision at full speed and the collision at rest since both paths lead to a collision with the pedestrian. If the velocity of the robot did not allow it to stop, the robot would then prefer to collide with

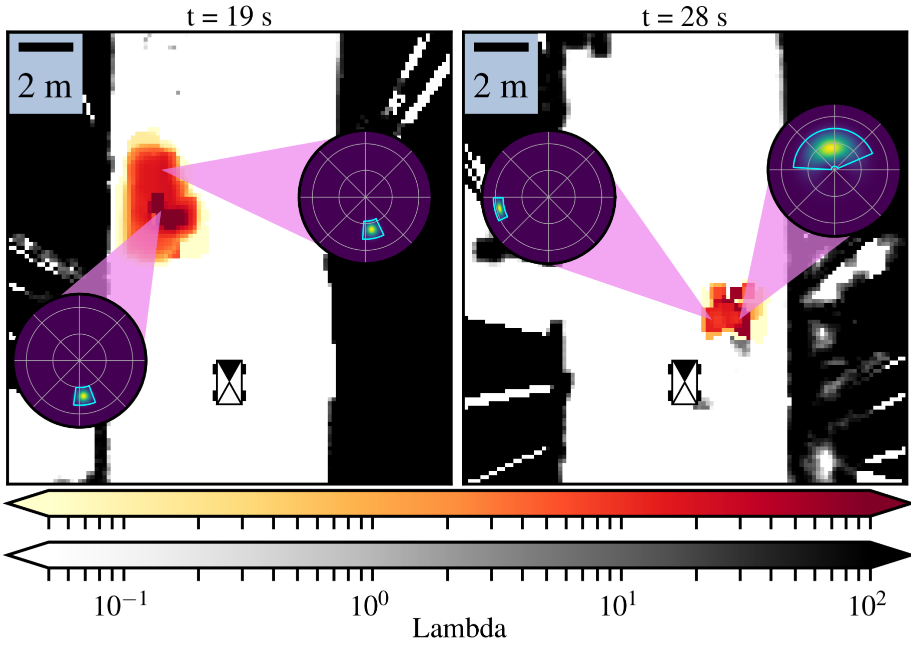


Figure 6.8: Example of field resulting from our method, where the robot is located at the bottom center of the map. The static (resp. dynamic) environment is represented using a gray (resp. red) scale. A car is approaching the robot at $t = 19\,\mathrm{s}$, whereas a pedestrian is crossing the road at $t = 28\,\mathrm{s}$ right in front of the robot. Some speed distributions of the cells are displayed in polar plots (angle for the orientation, radius for the velocity), where inner circles correspond to a step of $1\,\mathrm{m\,s^{-1}}$.

a parked car of same weight, as the resulting risk is lower. After a few iterations, the velocity of the pedestrian re-converged, and the robot continued on its way as soon as the pedestrian left. The robot accelerated to its maximum velocity and reached its goal safely.

6.4 Discussion

This extension to the dynamic world is still young and many enhancements can be done. As discussed in the introduction, the advantages of the Lambda-Fields only shine in situation of non-zero risk, meaning that the experimentations should reflect this as much as possible. Thus, experiments should be performed in situations of high-risk, which is not quite applicable as we would like to do more than one experiment which each robot. Coupling the framework with simulation programs such as CARLA [283] could provide the level of risk desired without actually breaking things. Other sensors, such as radars or cameras, could also be used to better inform the speed and classes of the obstacles.

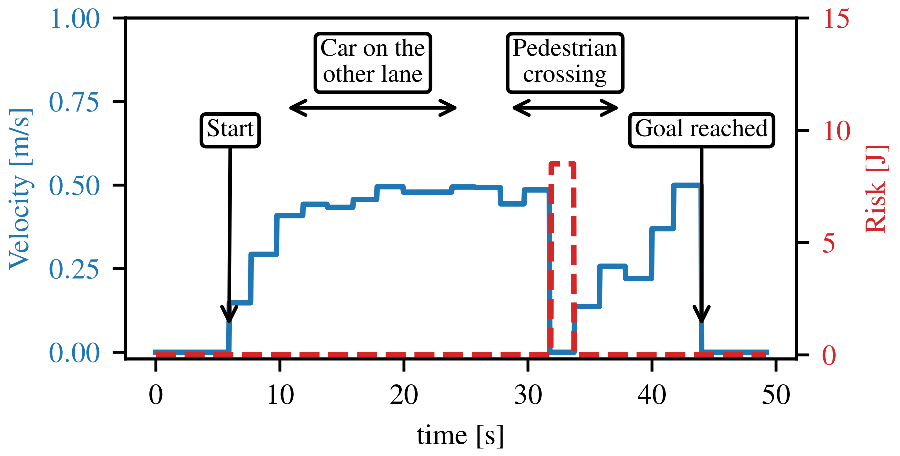


Figure 6.9: Risk undergone by the robot during its traversal with the associated speeds. First, a car passed the robot in the other lane, where its speed was precise enough not to cause the robot to brake or change direction. After that, a pedestrian emerges from behind the robot and crosses the road in front of it, leading the robot to stop and wait for the pedestrian to free the way. The robot then rejoins its goal without further obstacles.

Furthermore, the framework does not take into account the occlusions due to the environment. For instance, the framework does not handle the possibility that undetected cars can hide in the shadow zones of the crossroad. However, this can easily be added by adding random particles in the close unmeasured zones, but more an elegant solution can surely be developed. In the same context, a top-down approach could also be theorized to take into account the sensors of the infrastructure and of the other vehicles. Indeed, due to the possible high number of sensors available, it could be not possible to take them all into account in real-time. A framework that would be able to tell which sensor provides the information we seek for a safe travel (e.g., a camera looking at a blind spot of a crossroad) would be extremely valuable. Also, note that the top-down approach does not necessary discard information but instead provide an order of inquiries of the sensors (e.g., looking at the camera of the blind spot for two seconds may be more useful than spending time looking at the camera of the car next to us that sees the same information than us). Finally, more complex risk metrics can be included into the framework. An interesting idea is to have another framework feeding the Lambda-Field a risk metric that can change in real-time. Indeed, the robot might gain from changing the risk metric when transitioning from ultra-urban environments to rural ones.

6.5 Conclusion

In this chapter, we presented an extension of the Lambda-Field to the dynamic realm that is able to assess generic risks in occupancy grids. Using particles, we modeled both static and dynamic environments, deriving at the same time the nature of the obstacles.

We first showed how to compute the Dynamic Lambda-Field using lidar measurements. Then, the resulting dynamic map is used to assess the risk for a given path, here defined as the change of kinetic energy due to a collision with an obstacle. Using this formulation of the risk, the robot was able to plan real-time safe trajectories in a dynamic environment. More informed decisions were taken, where the robot could genuinely decide which decision is the best, either in an utilitarian or 'self-safety' way. The developed risk metric was here defined in a utilitarian way as the maximum of kinetic energy was taken and not only the kinetic energy change of the robot. As such, more hazardous paths for the robot can be taken in case where such a decision would save a pedestrian from a high-velocity impact.

As this work focused on the theoretical framework and providing a use case for its application, future work will involve extensive experiments with the framework in both real and simulated benchmarks, allowing for more complex and hazardous scenarios.

DISCUSSION AND PERSPECTIVES

In this part, we proposed a novel framework for risk assessment in occupancy grids. First, the framework was developed for static scenarios, allowing robot to cross unstructured environments such as tall grass. Then, we extended the framework to tackle dynamic obstacles in the context of urban navigation where the robot has to watch not only for its own safety but also the safety of the other agents. In contrast to the standard Bayesian occupancy grid, the Lambda-Fields provide a way to infer risks for given paths. One can understand the difference as the Bayesian occupancy grid is able to store and infer risks for a given position in the environment whereas Lambda-Fields infer risks for a given subset of the environment. As such, Lambda-Fields are better suited to quantify the risk of a path. The results of this thesis led to the publication of two conference articles [80], [129] and a journal article [284].

In a broader way, the Lambda-Field stores the intensity of a given event, that we defined throughout the whole thesis as the event 'collision'. Using this definition, we were able to derive a closed-form formula to build the Lambda-Field in real time with a lidar sensor. Then, the risk function has for generic purpose to quantify the risk of the event at a given configuration. First, we developed the core theory of the Lambda-Field in [129]. The framework was further extended in [284] to take into account unstructured obstacles and allow smooth navigation in tall grass. Finally, we enhanced the framework to take into account dynamic obstacles in [80], using a particle filter method. Hence, the framework is able to compute a map that allows generic risk assessment in both unstructured environments and dynamic, urban environments.

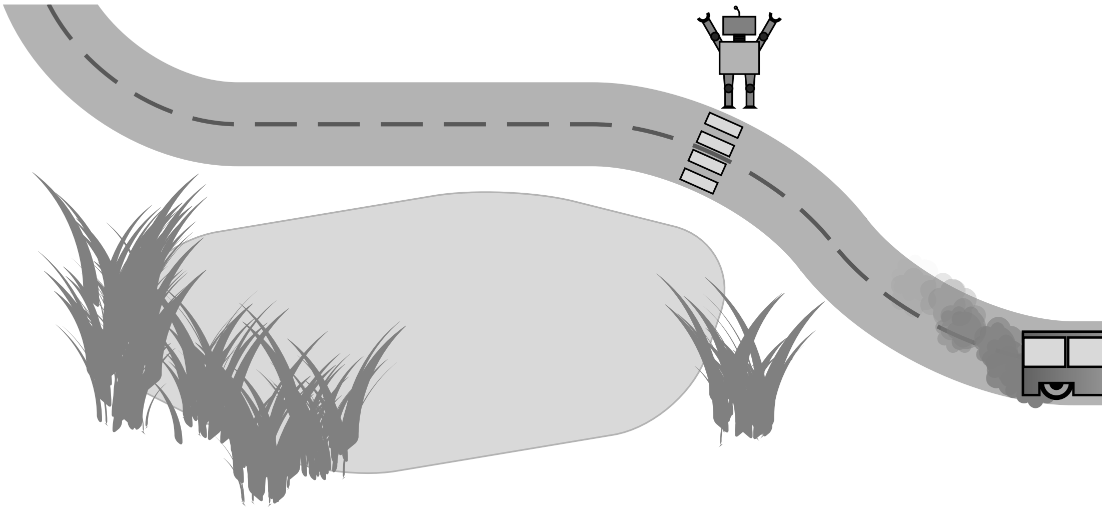
The risk function aimed in this thesis at quantifying the risk originating from the collision. However, the framework is generalizable for other types of events and other types of risks. For instance, we can define the event stored by the Lambda-Field as the event 'slippage', 'flat tire' or 'get stuck in loose soil'. Of course, the way the Lambda-Field is built has to be adapted, as for instance a lidar provides little information whether the robot will slip at this position. The risk function would also be adapted without difficulties, aiming at quantifying, for instance, the probability of loosing control of the vehicle. Current works are currently aiming at generalizing the framework in this direction, as done in [285] which defines the event as the deformation of the tires of the robot, leading to a more nuanced approach to collision assessment that allows for instance to safely climb road curbs.

Another direction is to actively manage the incompleteness of information. Sensors from other vehicles, as well as the ones mounted on the environment, could better inform the robot about incoming dangers. One typical example is the crossroads scenario, where the robot cannot see obstacles coming in the perpendicular roads. Cameras mounted on the environment, as well as an active communication between the vehicles, would effectively lift the uncertainty. As such, managing this incompleteness of the information would drastically improve the Lambda-Fields. Moreover, in the context of unstructured environments, sensing the obstacles with other sensors such as a radar would provide useful information. Indeed, not only a radar can sense obstacle, but also infer their properties to a certain extent. Thus, a radar could also infer the mass of the obstacles, conjointly with a camera. Another idea would be to equip the robot with an air blower. As low-mass obstacles, such as tall grass, would move under such forces, high-mass obstacles would not be impacted. These small, moving obstacles would then be easily detected using a lidar, thereby detecting which obstacles has a mass small enough to cross them.

Furthermore, one can note that the whole theory of the Lambda-Fields can be transposed to 3D. The sole issue is the computational requirements that indubitably increase as we add dimensions to the framework. As such, in the case where 3D Lambda-Fields are used in large environments, an octree representation of the environment would better fit the computational requirements than the simple tessellation we adopted throughout this thesis. However, in the case of manipulative robotics, the workspace is oftentimes relatively small and a constant tessellation size would suffice to map its entire environment. As such, the Lambda-Fields could be used without any modification to manipulative robotics. Indeed, as briefly mentioned in subsection 2.2.2, robotics arms share the same risk concepts than in mobile robotics, that is not to injure anyone during their displacements. As such, the Lambda-Fields provide a solution to the problem of risk analysis during path-planning, where the risk function can model the force of collision or the severity of the injury.

Another direction to take is a better path planning algorithm working directly on the Lambda-Fields. For now, the framework has been tested with clothoid curves (i.e., tentacles) but the robot is prone to fall into local minima and the trajectory is rather jerky. As discussed briefly in Section 5.5, there is still a lot of uncharted territory along this way. First, a consistent global path planning algorithm is needed in most applications and a more complex local planner could solve the aforementioned issues. The main problem to resolve is first how to assess the risk in a global planner. Indeed, the longer the path, the higher the risk. However, as humans, we do not angst more for a longer travel even though the overall risk is indubitably higher. In a local manner, we choose to define the risk as the risk of each decision, i.e., each command applied to the robot. As such, the higher the frequency of the commands, the ‘higher’ the overall risk of the travel is. This conclusion is natural in the way that the shorter you keep your eyes closed, the quicker you are to respond to danger and therefore the more risk you can withstand. Current works also tend to venture in this direction, which solve the eikonal equation (subsection 4.3.1) directly in the Lambda-Field.

In conclusion, we believe that following the aforementioned directions will lead to a robust, mathematically consistent framework that allows robots to safely roam the world.



Part III

Appendices

PUBLICATIONS ASSOCIATED WITH THIS THESIS

[80] J. Laconte, E. Randriamiarintsoa, A. Kasmi, F. Pomerleau, R. Chapuis, C. Debain, and R. Aufrère, "Dynamic Lambda-Field: A Counterpart of the Bayesian Occupancy Grid for Risk Assessment in Dynamic Environments," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[129] J. Laconte, C. Debain, R. Chapuis, F. Pomerleau, and R. Aufrere, "Lambda-Field: A Continuous Counterpart of the Bayesian Occupancy Grid for Risk Assessment," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 167–172, 2019.

[284] J. Laconte, A. Kasmi, F. Pomerleau, R. Chapuis, L. Malaterre, C. Debain, and R. Aufrère, "A Novel Occupancy Mapping Framework for Risk-Aware Path Planning in Unstructured Environments," *Sensors*, vol. 21, no. 22, p. 7562, 2021.

[285] J. Morceaux, J. Laconte, E. Randriamiarintsoa, T. Morell, L. Malaterre, D. Denis, R. Aufrere, and R. Chapuis, "Toward a Generalized Risk Assessment Method on Occupancy Grids," *Late Breaking Results Posters - 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

OTHER PUBLICATIONS DURING THE THESIS

[15] D. Baril, V. Grondin, S. P. Deschenes, J. Laconte, M. Vaidis, V. Kubelka, A. Gallant, P. Giguere, and F. Pomerleau, "Evaluation of Skid-Steering Kinematic Models for Subarctic Environments," *Proceedings - 2020 17th Conference on Computer and Robot Vision, CRV 2020*, pp. 198–205, 2020.

[37] M. Labussière, J. Laconte, and F. Pomerleau, "Geometry Preserving Sampling Method Based on Spectral Decomposition for Large-Scale Environments," *Frontiers in Robotics and AI*, vol. 7, 2020.

[145] A. Kasmi, J. Laconte, R. Aufrere, R. Theodose, D. Denis, and R. Chapuis, "An Information Driven Approach for Ego-Lane Detection Using Lidar and OpenStreetMap," in *16th IEEE International Conference on Control, Automation, Robotics and Vision, ICARCV 2020*, 2020, pp. 522–528.

[146] A. Kasmi, J. Laconte, R. Aufrere, D. Denis, and R. Chapuis, "End-to-End Probabilistic Ego-Vehicle Localization Framework," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 146–158, 2021.

[286] J. Laconte, S. P. Deschênes, M. Labussière, and F. Pomerleau, "Lidar measurement bias estimation via return waveform modelling in a context of 3D mapping," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, 2019, pp. 8100–8106.

[287] J. Laconte, A. Kasmi, F. Pomerleau, R. Chapuis, L. Malaterre, C. Debain, and R. Aufrère, "A Survey of Localization Methods for Autonomous Vehicles in Highway Scenarios," *Sensors*, 2021.

[288] M. Vaidis, J. Laconte, V. Kubelka, and F. Pomerleau, "Improving the Iterative Closest Point Algorithm using Lie Algebra," *IROS 2020 Workshop: Bringing geometric methods to robot learning, optimization and control*, 2020.

[289] D. Baril, S.-P. Deschênes, O. Gamache, M. Vaidis, D. LaRocque, J. Laconte, V. Kubelka, P. Giguère, and F. Pomerleau, "Kilometer-scale autonomous navigation in subarctic forests: Challenges and lessons learned," *Submitted to Field Robotics (FR)*, 2021.

Appendix A

Heterogeneous error regions

In the case that the error regions $\mathscr{E}_k$ have a different size for each lidar beam b_k, we need to further approximate the derivative of the log-likelihood. Under the same assumption that the h_i error regions $\mathscr{E}_k$ containing the cell c_i are small, we have

$$\begin{aligned}\frac{\partial \mathscr{L}(X|\lambda)}{\partial \lambda_i} &= -m_c \cdot \Delta a + \sum_{k=0}^{h_i-1} \frac{\Delta a}{\exp(e_k \lambda_i) - 1} \\ &\approx -m_c \cdot \Delta a + \sum_{k=0}^{h_i-1} \frac{\Delta a}{e_k \lambda_i},\end{aligned} \tag{A.1}$$

leading to

$$\lambda_i = \frac{1}{m_i} \sum_{k=0}^{h_i-1} \frac{1}{e_k}. \tag{A.2}$$

This approximation over-estimates the lambdas compared to Equation 5.23. Indeed, in the special case where all the $\mathscr{E}_k$ have the same area e, the computed lambdas from Equation A.2 are

$$\lambda_i = \frac{1}{e} \frac{h_i}{m_i}. \tag{A.3}$$

As $\forall x \in \mathbb{R}_{\geq 0}$, $x \geq \ln(1+x)$, we will always over-estimate the lambdas using Equation A.2. This is the desired behavior as under-estimating the lambdas would lead to under-estimate the risk.

APPENDIX B

PROOF OF EQUATION 5.41

We here prove Equation 5.41. We have two random variables: the area at which the robot collides A and the mass M of the cell where the collision happened, of marginal probability density functions $f(\cdot)$ and $f^m(\cdot)$. Note that we do not have directly access to $f^m(\cdot)$ but only $f_i^m(\cdot) = f^m(\cdot|\Delta a i)$, the probability density function of the mass given where the collision happened. Under the assumption that the risk $r(\cdot)$ is constant inside each cell, the expectation of the function $r(A,M)$ is

$$\begin{aligned}
&\mathbb{E}\left[r(A,M)\right] = \int_0^{N\Delta a}\int_0^{\infty} r_m(a,m) f(a) f^m(m|a)\,\mathrm{d}m\,\mathrm{d}a \\
&= \sum_{i=0}^{N-1}\int_{i\Delta a}^{(i+1)\Delta a} f(a)\int_0^{\infty} r_m(a,m) f_i^m(m)\,\mathrm{d}m\,\mathrm{d}a \\
&= \sum_{i=0}^{N-1}\left[\int_{i\Delta a}^{(i+1)\Delta a} f(a)\,\mathrm{d}a\right]\left[\int_0^{\infty} r_m(\Delta a i, m) f_i^m(m)\,\mathrm{d}m\right] \\
&= \sum_{i=0}^{N-1} K_i \int_0^{\infty} r_m(\Delta a i, m_i) f_i^m(m_i)\,\mathrm{d}m \\
&= \sum_{i=0}^{N-1} K_i \sum_{k=0}^{\infty} \alpha_{ik} r_m(\Delta a i, k\Delta_m),
\end{aligned} \tag{B.1}$$

for a path $\mathscr{P}$ going through the cells $\{c_i\}_{0:N-1}$, and with

$$K_i = \exp\left(-\Lambda_m(\{c_j\}_{0:i-1})\right)\left[1-\exp\left(-\Lambda_m(\{c_i\})\right)\right]. \tag{B.2}$$

APPENDIX C

PROBABILISTIC ERROR REGION

If the range sensor has a large error zone, the inflation of the obstacles may become problematic. Therefore, we give here a way to estimate the Lambda-Field using a probabilistic error region. Let the error region $\mathscr{E} : \mathbb{R} \to \mathscr{P}(\mathbb{R}^2)$ an application that takes a parameter and returns a subspace of $\mathbb{R}^2$. One example is the application that gives from the radius r the ball centered on the lidar measurement $\mathbf{x_l} \in \mathbb{R}^2$: $\{\mathbf{x} \in \mathbb{R}^2, |\mathbf{x} - \mathbf{x_l}| \leq r\}$. Furthermore, let σ be a random variable of probability density function $f_\sigma(\cdot)$ and a Lambda-Field $\lambda\colon \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$. Under these considerations, the expectation of the intensity of the error zone $\mathscr{E}(\sigma)$ is

$$\begin{aligned}\mathbb{E}\left[\int_{\mathscr{E}(\sigma)} \lambda(\mathbf{x})\,\mathrm{d}\mathbf{x}\right] &= \int_{\mathbb{R}} f_\sigma(s) \int_{\mathscr{E}(s)} \lambda(\mathbf{x})\,\mathrm{d}\mathbf{x}\,\mathrm{d}s \\ &= \int_{\mathbb{R}} f_\sigma(s) \int_{\mathbb{R}^2} \lambda(\mathbf{x}) \cdot \mathbf{1}_{\mathscr{E}(s)}(\mathbf{x})\,\mathrm{d}\mathbf{x}\,\mathrm{d}s.\end{aligned} \tag{C.1}$$

Under the assumption that the expectation of the intensity of the error zone is finite, we can switch the integration order using Fubini's theorem and find a more convenient form:

$$\begin{aligned}\mathbb{E}\left[\int_{\mathscr{E}(\sigma)} \lambda(\mathbf{x})\,\mathrm{d}\mathbf{x}\right] &= \int_{\mathbb{R}^2} \lambda(\mathbf{x}) \int_{\mathbb{R}} f_\sigma(s) \cdot \mathbf{1}_{\mathscr{E}(s)}(\mathbf{x})\,\mathrm{d}s\,\mathrm{d}\mathbf{x} \\ &= \int_{\mathbb{R}^2} \lambda(\mathbf{x})\,\mathbb{P}(\mathbf{x} \in \mathscr{E}(\sigma))\,\mathrm{d}\mathbf{x},\end{aligned} \tag{C.2}$$

where $\mathbf{1}_X$ is the identity operator, i.e., $\mathbf{1}_X(\mathbf{x}) = 1$ if $\mathbf{x} \in X$ and 0 otherwise. Using this expectation as the new intensity function $\Lambda(\mathscr{C})$ and putting it back into Equation 5.20, the lambdas are now estimated using the same counters h_i and m_i that now represent respectively the sum of the probabilities of being in the error zone and the sum of the probabilities of not being in the error zone of each lidar measurement. One can note that in the case where the error region is known, meaning that $\mathbb{P}(x \in \mathscr{E}(\sigma))$ is either equal to zero or one, we fall back on the previously derived equations as h_i and m_i regain their function of counting the number of times the cell has been or not in the error region.

APPENDIX D

COMPUTATION OF THE CONFIDENCE INTERVAL OF THE RISK

In this thesis, the upper risk is computed using the upper bound of the confidence interval of each lambda. However, taking these values may not lead to the maximum risk. Indeed, if a robot goes through a homogeneous field (constant lambda for all the field) while accelerating, taking the upper bound leads to a lesser risk. If a collision happens fast (high lambda), the risk will be low as the robot has not the time to accelerate.

Hence, for a path crossing the cells $\{c_i\}$, we need to find the set of lambdas $\{\lambda_i \in [\lambda_{Li}, \lambda_{Ui}]\}$ that maximize the expectation of collision (or minimize for the lower bound).

We have

$$\begin{aligned}
\frac{\partial}{\partial \lambda_i}\mathbb{E}[r(A)] &= r(\Delta a i) a_i \Delta a \exp\left(-\Delta a \lambda_i\right) - \sum_{k>i} r(\Delta a k)\Delta a a_k c_k \\
&= r(\Delta a i) a_i \left[\Delta a \exp\left(-\Delta a \lambda_i\right) - \Delta a \sum_{k>i} \frac{r(\Delta a k)}{r(\Delta a i)} \frac{a_k}{a_{i+1}} \exp\left(-\Delta a \lambda_i\right) c_k\right] \qquad \text{(D.1)} \\
&= \underbrace{a_i \Delta a \exp\left(-\Delta a \lambda_i\right) r(\Delta a i)}_{>0} \left(1 - \sum_{k>i} \frac{r(\Delta a k)}{r(\Delta a i)} \frac{a_k}{a_{i+1}} c_k\right).
\end{aligned}$$

with

$$\begin{aligned}
a_i &= \exp\left(-\Delta a \sum_{j<i} \lambda_j\right) \qquad \text{and} \\
c_i &= 1 - \exp\left(-\Delta a \lambda_i\right).
\end{aligned}$$

As we can see, the derivative is either $0 \,\forall \lambda_i \in \mathbb{R}_{\geq 0}$ or does not nullify as the second term does not depend on λ_i. Hence, the maximum of the function is at the border, i.e., λ_{Li} or λ_{Ui} (in the special case where the derivate is null for all lambda, the two values lead to the same risk). The maximum of the expectation is at

$$\lambda_i = \lambda_{Ui} \text{ if } \left(1 - \sum_{k>i} \frac{r(\Delta a k)}{r(\Delta a i)} \frac{a_k}{a_{i+1}} c_k\right) \geq 0$$

$$\lambda_i = \lambda_{Li} \text{ otherwise}$$

and the minimum is at the opposite condition.

First, we see that in the case where $r(\cdot)$ is constant, the derivative is always positive, hence leading to λ_{Ui} all the time, which is intuitively correct.

However, the computation of the condition is not easy as it depends of the other lambdas that we need to find. To compute these values, we need to point out the fact that the condition for λ_i only depends on $\{\lambda_k\}_{k>i}$. More specifically, for a path going through the cells $\{c_i\}_{0\leq i<N}$, the derivative for λ_{N-1} is

$$\Delta a \exp\left(-\Delta a \lambda_{N-1}\right) > 0, \tag{D.2}$$

meaning that the last lambda will always be $\lambda_{U(N-1)}$. Using the fact that the condition on λ_{N-2} only depends on λ_{N-1} which is known, we are able to compute its value leading to the maximum of risk (or minimum). Using this recurrence, we are able to compute one by one the lambdas from λ_{N-1} to λ_0.

BIBLIOGRAPHY

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* MIT Press, 2005.

[2] A. Dargazany and K. Berns, "Terrain Traversability Analysis using Organized Point Cloud , Superpixel Surface Normals-based segmentation and PCA-based Classification," *Workshop on Field and Assistive Robotics (WFAR); Lahore, Pakistan*, 2014.

[3] P. Papadakis, "Terrain traversability analysis methods for unmanned ground vehicles: A survey," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1373–1385, 2013.

[4] J. Leonard, J. Cort, and T. Sim, "Sampling-based Path Planning on Configuration-Space Costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.

[5] I. S. Kweon and T. Kanade, "High-Resolution Terrain Map from Multiple Sensor Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 278–292, 1992.

[6] M. Daily, J. Harris, D. Keirsey, D. Olin, D. Payton, K. Reiser, J. Rosenblatt, D. Tseng, and V. Wong, "Autonomous cross-country navigation with the ALV," *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 718–726, 1988.

[7] K. E. Olin and D. Y. Tseng, "Autonomous Cross-Country Navigation An Integrated Perception and Planning System," *IEEE Expert-Intelligent Systems and their Applications*, vol. 6, no. 4, pp. 16–30, 1991.

[8] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[9] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[10] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 3231–3237, 2005.

[11] N. A. Melchior and R. Simmons, "Particle RRT for path planning with uncertainty," *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 1617–1624, 2007.

[12] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness, “Towards Learned Traversability for Robot Navigation: From Underfoot to the Far Field,” *Journal of Field Robotics*, vol. 23, no. 11-12, pp. 1005–1017, 2006.

[13] J. G. Mooney and E. N. Johnson, “A Comparison of Automatic Nap-of-the-earth Guidance Strategies for Helicopters,” *Journal of Field Robotics*, vol. 33, no. 1, pp. 1–17, 2014.

[14] R. Enain, B. Thuilot, C. Cariou, and P. Martinet, “Mixed Kinematic and Dynamic Sideslip Angle Observer for Accurate Control of Fast Off-Road Mobile Robots,” *Journal of Field Robotics*, vol. 27, no. 2, pp. 181–196, 2010.

[15] D. Baril, V. Grondin, S. P. Deschenes, J. Laconte, M. Vaidis, V. Kubelka, A. Gallant, P. Giguere, and F. Pomerleau, “Evaluation of Skid-Steering Kinematic Models for Subarctic Environments,” *Proceedings - 2020 17th Conference on Computer and Robot Vision, CRV 2020*, pp. 198–205, 2020.

[16] C. Cunningham, M. Ono, I. Nesnas, J. Yen, and W. L. Whittaker, “Locally-adaptive slip prediction for planetary rovers using Gaussian processes,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5487–5494, 2017.

[17] M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. S. Albus, “Learning traversability models for autonomous mobile vehicles,” *Autonomous Robots*, vol. 24, no. 1, pp. 69–86, 2008.

[18] C. Brooks, K. Iagnemma, and S. Dubowsky, “Vibration-based terrain analysis for mobile robots,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, 2005, pp. 3415–3420.

[19] Carlton C. Allen, R. V. Morris, K. M. Jager, D. C. Golden, D. J. Lindstrom, M. Lindstrom, and J. P. Lockwood, “Martian Regolith Simulant JSC Mars-1,” *Lunar and Planetary Science XXIX*, no. Table 2, pp. 4–5, 2004.

[20] F. L. Garcia Bermudez, R. C. Julian, D. W. Haldane, P. Abbeel, and R. S. Fearing, “Performance analysis and terrain classification for a legged robot over rough terrain,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 513–519, 2012.

[21] E. Coyle and E. G. Collins, “A Comparison of Classifier Performance for Vibration-Based Terrain Classification,” *Army Science Conference*, pp. 1–4, 2008.

[22] L. Ojeda, J. Borenstein, G. Witus, and R. Karlsen, “Terrain Characterization and Classification with a Mobile Robot,” *Journal of Field Robotics*, vol. 23, no. 2, pp. 103–122, 2006.

[23] A. Angelova, L. Matthies, D. Helmick, and P. Perona, “Learning and Prediction of Slip from Visual Information,” *Journal of Field Robotics*, vol. 24, no. 3, pp. 205–231, 2007.

[24] C. Wellington and A. Stentz, "Online adaptive rough-terrain navigation in vegetation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2004, no. 1, pp. 96–101, 2004.

[25] I. M. Leppänen, P. J. Virekoski, and A. J. Halme, "Sensing terrain parameters and the characteristics of vehicle-terrain interaction using the multimode locomotion system of a robot," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 500–505, 2008.

[26] M. Bajracharya, A. Howard, L. H. Matthies, B. Tang, and M. Turmon, "Autonomous Off-Road Navigation with End-to-End Learning for the LAGR Program," *Journal of Field Robotics*, vol. 26, no. 1, pp. 3–25, 2009.

[27] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness, "Towards Learned Traversability for Robot Navigation: From Underfoot to the Far Field," *Journal of Field Robotics*, vol. 23, no. 11-12, pp. 1005–1017, 2006.

[28] R. Hoffman and E. Krotkov, "Terrain Roughness Measurement from Elevation Maps," *Mobile Robots IV*, vol. 1195, p. 104, 1990.

[29] L. Lu, C. Ordonez, E. G. Collins, and E. M. DuPont, "Terrain surface classification for autonomous ground vehicles using a 2D laser stripe-based structured light sensor," *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 2174–2181, 2009.

[30] L. Lu, C. Ordonez, E. G. Collins, E. Coyle, and D. Palejiya, "Terrain surface classification with a control mode update rule using a 2D laser stripe-based structured light sensor," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 954–965, 2011.

[31] D. K. Pai and L. M. Reissell, "Multiresolution rough terrain motion planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 19–33, 1998.

[32] D. Langer, J. K. Rosenblatt, and M. Hebert, "A Behavior-Based System for Off-Road Navigation," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 776–783, 1994.

[33] D. B. Gennery, "Traversability analysis and path planning for a planetary rover," *Autonomous Robots*, vol. 6, no. 2, pp. 131–146, 1999.

[34] C. Ye, "Navigating a mobile robot by a traversability field histogram," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 361–372, 2007.

[35] J. C. Andersen, M. R. Blas, O. Ravn, N. A. Andersen, and M. Blanke, "Traversable terrain classification for outdoor autonomous robots using single 2D laser scans," *Integrated Computer-Aided Engineering*, vol. 13, no. 3, pp. 223–232, 2006.

[36] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr, “Recent progress in local and global traversability for planetary rovers,” *Proceedings-IEEE International Conference on Robotics and Automation*, vol. 2, no. April, pp. 1194–1200, 2000.

[37] M. Labussière, J. Laconte, and F. Pomerleau, “Geometry Preserving Sampling Method Based on Spectral Decomposition for Large-Scale Environments,” *Frontiers in Robotics and AI*, vol. 7, 2020.

[38] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, “Natural terrain classification using three-dimensional ladar data for ground robot mobility,” *Journal of Field Robotics*, vol. 23, no. 10, pp. 839–861, 2006.

[39] G. Medioni, M.-S. Lee, and C.-K. Tang, *A computational framework for segmentation and grouping*. Elsevier, 2000.

[40] N. Heckman, J. F. Lalonde, N. Vandapel, and M. Hebert, “Potential negative obstacle detection by occlusion labeling,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 2168–2173, 2007.

[41] S. B. Goldberg, M. W. Maimone, and L. Matthies, “Stereo vision and rover navigation software for planetary exploration,” *IEEE Aerospace Conference Proceedings*, vol. 5, pp. 2025–2036, 2002.

[42] J. Larson, M. Trivedi, and M. Bruch, “Off-Road Terrain Traversability Analysis and Hazard Avoidance for UGVs,” *Ifac*, pp. 1–7, 2011.

[43] J. Larson and M. Trivedi, “Lidar based off-road negative obstacle detection and analysis,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 192–197, 2011.

[44] L. Chen, J. Yang, and H. Kong, “Lidar-histogram for fast road and obstacle detection,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1343–1348, 2017.

[45] A. Murarka, M. Sridharan, and B. Kuipers, “Detecting obstacles and drop-offs using stereo and motion cues for safe local motion,” *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 702–708, 2008.

[46] T. Siméon, “Motion planning for a non-holonomic mobile robot on 3-dimensional terrains,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 708 LNCS P, no. 91, pp. 38–50, 1993.

[47] T. Simeon and B. Dacre-Wright, “Practical motion planner for all-terrain mobile robots,” *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, pp. 1357–1363, 1993.

[48] B. D. Wright and T. Simeon, “Free space representation for a mobile robot moving on a rough terrain,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, pp. 37–43, 1993.

[49] D. Bonnafous, S. Lacroix, and T. Siméon, "Motion generation for a rover on rough terrains," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2, 2001, pp. 784–789.

[50] T. Kubota, Y. Kuroda, Y. Kunii, and T. Yoshimitsu, "Path Planning for Newly Developed Microrover," *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, no. 4, pp. 3710–3715, 2001.

[51] N. Vandapel, R. R. Donamukkala, and M. Hebert, "Unmanned ground vehicle navigation using aerial ladar data," *International Journal of Robotics Research*, vol. 25, no. 1, pp. 31–51, 2006.

[52] G. Ishigami, K. Nagatani, and K. Yoshida, "Path planning and evaluation for planetary rovers based on dynamic mobility index," *IEEE International Conference on Intelligent Robots and Systems*, pp. 601–606, 2011.

[53] P. Papadakis and F. Pirri, "3D mobility learning and regression of articulated, tracked robotic vehicles by physics-based optimization," *VRIPHYS 2012 - 9th Workshop on Virtual Reality Interactions and Physical Simulations*, pp. 147–156, 2012.

[54] G.-J. M. Kruijff, F. Pirri, M. Gianni, P. Papadakis, M. Pizzoli, A. Sinha, V. Tretyakov, T. Linder, E. Pianese, S. Corrao, *et al.*, "Rescue robots at earthquake-hit Mirandola, Italy: A field report," *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2012*, pp. 1–8, 2012.

[55] M. Norouzi, J. V. Miro, and G. Dissanayake, "Planning Stable and Efficient Paths for Reconfigurable Robots On Uneven Terrain," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 87, no. 2, pp. 291–312, 2017.

[56] A. Stentz, "Focussed_Dstar_Ijcai95.Pdf," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 95, no. August, pp. 1652–1659, 1995.

[57] A. Kelly, "An intelligent, predictive control approach to the high-speed cross-country autonomous navigation problem," *Engineering*, 1995.

[58] D. Wettergreen, P. Tompkins, C. Urmson, M. Wagner, and W. Whittaker, "Sun-synchronous robotic exploration: Technical description and field experimentation," *International Journal of Robotics Research*, vol. 24, no. 1, pp. 3–30, 2005.

[59] D. Helmick, A. Angelova, and L. Matthies, "Terrain adaptive navigation for planetary rovers," *Journal of Field Robotics*, vol. 26, no. 4, pp. 391–410, 2009.

[60] T. Huntsberger, A. Jain, J. Cameron, G. Woodward, D. Myers, and G. Sohl, "Characterization of the ROAMS simulation environment for testing rover mobility on sloped terrain," *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2008.

[61] S. Thrun, M. Montemerlo, and A. Aron, "Probabilistic terrain analysis for high-speed desert driving," *Robotics: Science and Systems*, vol. 2, pp. 161–167, 2007.

[62] G. Dubbelman, W. Van Der Mark, J. C. Van Den Heuvel, and F. C. Groen, "Obstacle detection during day and night conditions using stereo vision," *IEEE International Conference on Intelligent Robots and Systems*, pp. 109–116, 2007.

[63] S. Kuthirummal, A. Das, and S. Samarasekera, "A graph traversal based algorithm for obstacle detection using lidar or stereo," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3874–3880, 2011.

[64] M. Bellone, G. Reina, N. I. Giannoccaro, and L. Spedicato, "3D traversability awareness for rough terrain mobile robots," *Sensor Review*, vol. 34, no. 2, pp. 220–232, 2014.

[65] M. Montemerlo and S. Thrun, "A multi-resolution pyramid for outdoor robot terrain perception," *Proceedings of the National Conference on Artificial Intelligence*, pp. 464–469, 2004.

[66] D. Ferguson, A. Morris, D. Hähnel, C. Baker, Z. Omohundro, C. Reverte, S. Thayer, C. Whittaker, W. Whittaker, W. Burgard, and S. Thrun, "An autonomous robotic system for mapping abandoned mines," *Advances in Neural Information Processing Systems*, no. Ml, pp. 587–594, 2004.

[67] D. Joho, C. Stachniss, P. Pfaff, and W. Burgard, "Autonomous exploration for 3D map learning," *Informatik aktuell*, pp. 22–28, 2007.

[68] V. Molino, R. Madhavan, E. Messina, A. Downs, S. Balakirsky, and A. Jacoff, "Traversability metrics for rough terrain applied to repeatable test methods," in *IEEE International Conference on Intelligent Robots and Systems*, 2007, pp. 1787–1794.

[69] A. Andrakhanov and A. Stuchkov, "Traversability estimation system for mobile robot in heterogeneous environment with different underlying surface characteristics," *Proceedings of the 12th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2017*, vol. 1, pp. 549–554, 2017.

[70] Y. Guo, A. Song, J. Bao, and H. Zhang, "Optimal path planning in field based on traversability prediction for mobile robot," *2011 International Conference on Electric Information and Control Engineering, ICEICE 2011 - Proceedings*, pp. 563–566, 2011.

[71] A. Howard, H. Seraji, and E. Tunstel, "A rule-based Fuzzy Traversability Index for mobile robot navigation," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3067–3071, 2001.

[72] A. Howard and H. Seraji, "Vision-based terrain characterization and traversability assessment," *Journal of Robotic Systems*, vol. 18, no. 10, pp. 577–587, 2001.

[73] D. Kim, S. M. Oh, and J. M. Rehg, "Traversability classification for UGV navigation: A comparison of patch and superpixel representations," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3166–3173, 2007.

[74] P. Filitchkin and K. Byl, "Feature-based terrain classification for LittleDog," *IEEE International Conference on Intelligent Robots and Systems*, no. 2, pp. 1387–1392, 2012.

[75] Y. N. Khan, P. Komma, and A. Zell, "High resolution visual terrain classification for outdoor robots," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1014–1021, 2011.

[76] F. Yandun Narváez, E. Gregorio, A. Escolà, J. R. Rosell-Polo, M. Torres-Torriti, and F. Auat Cheein, "Terrain classification using ToF sensors for the enhancement of agricultural machinery traversability," *Journal of Terramechanics*, vol. 76, pp. 1–13, 2018.

[77] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Image classification for ground traversability estimation in robotics," *International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 325–336, 2017.

[78] C. Caborni, S. Y. Ko, E. De Momi, G. Ferrigno, and F. R. Y Baena, "Risk-based path planning for a steerable flexible probe for neurosurgical intervention," *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 866–871, 2012.

[79] M. Vaillant, C. Davatzikos, R. H. Taylor, and R. Nick Bryan, "A path-planning algorithm for image-guided neurosurgery," *CVRMed-MRCAS'97*, pp. 467–476, 1997.

[80] J. Laconte, E. Randriamiarintsoa, A. Kasmi, F. Pomerleau, R. Chapuis, C. Debain, and R. Aufrère, "Dynamic Lambda-Field: A Counterpart of the Bayesian Occupancy Grid for Risk Assessment in Dynamic Environments," *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[81] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Soft-tissue injury in robotics," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3426–3433, 2010.

[82] T. Fraichard and H. Asama, "Inevitable Collision States a Step Towards Safer Robots?" *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, no. 10, pp. 388–393, 2003.

[83] T. Fraichard, "A short paper about motion safety," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1140–1145, 2007.

[84] M. Brännström, E. Coelingh, and J. Sjöberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 658–669, 2010.

[85] R. Labayrade, C. Royere, and D. Aubert, "Experimental assessment of the RESCUE collision-mitigation system," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 1, pp. 89–102, 2007.

[86] S. Gim, L. Adouane, S. Lee, and J. P. Dérutin, "Clothoids Composition Method for Smooth Path Generation of Car-Like Vehicle Navigation," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 88, no. 1, pp. 129–146, 2017.

[87] J. Hou, G. F. List, and X. Guo, "New algorithms for computing the time-to-collision in freeway traffic simulation models," *Computational Intelligence and Neuroscience*, vol. 2014, 2014.

[88] C. Pek and M. Althoff, "Computationally Efficient Fail-safe Trajectory Planning for Self-driving Vehicles Using Convex Optimization," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, pp. 1447–1454, 2018.

[89] A. V. Savkin and C. Wang, "Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1568–1580, Oct. 2014.

[90] C. G. Keller, T. Dang, H. Fritz, A. Joos, C. Rabe, and D. M. Gavrila, "Active pedestrian safety by automatic braking and evasive steering," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1292–1304, 2014.

[91] S. Noh and K. An, "Decision-Making Framework for Automated Driving in Highway Environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 58–71, 2018.

[92] J. C. Hayward, "Near-Miss Determination Through," *Highway Research Board*, pp. 24–35, 1972.

[93] C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Yong, J. D. Yoder, C. Tay, K. Mekhnacha, and A. Nègre, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 4, pp. 4–19, 2011.

[94] D. N. Lee, "A theory of visual control of braking based on information about time–to–collision," *Perception*, vol. 5, no. 4, pp. 437–459, 1976.

[95] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis and Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[96] F. Jiménez, J. E. Naranjo, and F. García, "An Improved Method to Calculate the Time-to-Collision of Two Vehicles," *International Journal of Intelligent Transportation Systems Research*, vol. 11, no. 1, pp. 34–42, 2013.

[97] J. R. Ward, G. Agamennoni, S. Worrall, A. Bender, and E. Nebot, "Extending Time to Collision for probabilistic reasoning in general traffic scenarios," *Transportation Research Part C: Emerging Technologies*, vol. 51, pp. 66–82, 2015.

[98] B. L. Allen, B. T. Shin, and P. Cooper, "Analysis of Traffic Conflicts and Collisions," *Transportation Research Record*, no. 667, pp. 67–74, 1978.

[99] C. Hupfer, "Deceleration to safety time (DST)-a useful figure to evaluate traffic safety," *ICTCT Conference Proceedings of Seminar*, vol. 3, pp. 5–7, 1997.

[100] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A multilevel collision mitigation approach - Its situation assessment, decision making, and performance tradeoffs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 528–540, 2006.

[101] D. Richards, "Relationship between Speed and Risk of Fatal Injury: Pedestrians and Car Occupants," *Road Safety Web Publication*, no. 16, pp. 19–22, 2010.

[102] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, 2016.

[103] R. Benenson, T. Fraichard, and M. Parent, "Achievable safety of driverless ground vehicles," *2008 10th International Conference on Control, Automation, Robotics and Vision, ICARCV 2008*, pp. 515–521, 2008.

[104] S. Bouraine, T. Fraichard, and H. Salhi, "Relaxing the inevitable collision state concept to address provably safe mobile robot navigation with limited field-of-views in unknown dynamic environments," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2985–2991, 2011.

[105] L. A. S. Guardini, A. Spalanzani, C. Laugier, P. Martinet, A. L. Do, and T. Hermitte, "Employing Severity of Injury to Contextualize Complex Risk Mitigation Scenarios," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1839–1845, 2020.

[106] F. Sandblom and M. Brännström, "Probabilistic threat assessment and driver modeling in collision avoidance systems," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 914–919, 2011.

[107] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[108] A. Berthelot, A. Tamke, T. Dang, and G. Breuel, "Handling uncertainties in criticality assessment," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 571–576, 2011.

[109] J. Eggert, "Predictive risk estimation for intelligent ADAS functions," *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014*, pp. 711–718, 2014.

[110] C. Fulgenzi, A. Spalanzani, and C. Laugier, “Probabilistic motion planning among moving obstacles following typical motion patterns,” *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 4027–4033, 2009.

[111] L. Wang, C. F. Lopez, and C. Stiller, “Realistic Single-Shot and Long-Term Collision Risk for a Human-Style Safer Driving,” *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 2073–2080, 2020.

[112] F. Oboril and K. U. Scholl, “Risk-Aware Safety Layer for AV Behavior Planning,” *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 1922–1928, 2020.

[113] J. Rios-Martinez, A. Spalanzani, and C. Laugier, “Understanding human interaction for probabilistic autonomous navigation using risk-RRT approach,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 2014–2019, 2011.

[114] O. Ogorodnikova, “A fuzzy theory in the risk assessment and reduction algorithms for a human centered robotics,” in *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, IEEE, 2009, pp. 340–345.

[115] D. Maier, M. Bennewitz, and C. Stachniss, “Self-supervised obstacle detection for humanoid navigation using monocular vision and sparse laser data,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1263–1269, 2011.

[116] A. Fabisch, C. Petzoldt, M. Otto, and F. Kirchner, “A Survey of Behavior Learning Applications in Robotics–State of the Art and Perspectives,” *arXiv preprint arXiv:1906.01868*, 2019.

[117] I. Kostavelis, L. Nalpantidis, and A. Gasteratos, “Collision risk assessment for autonomous robots by offline traversability learning,” *Robotics and Autonomous Systems*, vol. 60, no. 11, pp. 1367–1376, 2012.

[118] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, “Collision detection for industrial collaborative robots: A deep learning approach,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.

[119] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, “Demonstration based trajectory optimization for generalizable robot motions,” *IEEE-RAS International Conference on Humanoid Robots*, pp. 515–522, 2016.

[120] T. Malm, J. Viitaniemi, J. Latokartano, S. Lind, O. Venho-Ahonen, and J. Schabel, “Safety of interactive robotics-learning from accidents,” *International Journal of Social Robotics*, vol. 2, no. 3, pp. 221–227, 2010.

[121] A. Terra, H. Riaz, K. Raizer, A. Hata, and R. Inam, “Safety vs. Efficiency: AI-Based Risk Mitigation in Collaborative Robotics,” *2020 6th International Conference on Control, Automation and Robotics, ICCAR 2020*, pp. 151–160, 2020.

[122] S. Feyzabadi and S. Carpin, "Risk-aware path planning using hirerachical constrained Markov Decision Processes," *IEEE International Conference on Automation Science and Engineering*, pp. 297–303, 2014.

[123] S. Martin, L. Murphy, and P. Corke, "Building Large Scale Traversability Maps Using Vehicle Experience," *The 13th International Symposium on Experimental Robotics (ISER)*, vol. 88, pp. 891–905, 2013.

[124] G. Sakayori and G. Ishigami, "Energy efficient slope traversability planning for mobile robot in loose soil," *Proceedings - 2017 IEEE International Conference on Mechatronics, ICM 2017*, pp. 99–104, 2017.

[125] J. Bentham, *An Introduction to the Principles of Morals and Legislation.* 1789.

[126] J. S. Mill, *Utilitarianism.* 1859.

[127] S. Karnouskos, "The role of utilitarianism, self-safety, and technology in the acceptance of self-driving cars," *Cognition, Technology and Work*, 2020.

[128] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," *arXiv*, pp. 1–37, 2017.

[129] J. Laconte, C. Debain, R. Chapuis, F. Pomerleau, and R. Aufrere, "Lambda-Field: A Continuous Counterpart of the Bayesian Occupancy Grid for Risk Assessment," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 167–172, 2019.

[130] A. Majumdar and M. Pavone, "How should a robot assess risk? Towards an axiomatic theory of risk in robotics," *arXiv*, pp. 75–84, 2017.

[131] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 267–283, 2012.

[132] K. Macek, D. Alejandro, V. Govea, T. Fraichard, K. Macek, D. Alejandro, V. Govea, T. Fraichard, and R. S. To-, "Towards Safe Vehicle Navigation in Dynamic Urban Scenarios To cite this version : Towards Safe Vehicle Navigation in Dynamic Urban Scenarios," *Automatika–Journal for Control, Measurement, Electronics, Computing and Communications*, 2009.

[133] M. Sivak and B. Schoettle, "Road Safety With Self-Driving Vehicles: General Limitations and road sharing with conventional vehicles," *Transportation Research Institute*, no. January, p. 11, 2015.

[134] P. Junietz, W. Wachenfeld, K. Klonecki, and H. Winner, "Evaluation of Different Approaches to Address Safety Validation of Automated Driving," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 491–496, 2018.

[135] A. C. Serban, E. Poll, and J. Visser, "Tactical safety reasoning. A case for autonomous vehicles," *IEEE Vehicular Technology Conference*, vol. 2018-June, no. 2, pp. 1–5, 2018.

[136] P. Ertle, H. Voos, and D. Söffker, "On risk formalization of on-line risk assessment for safe decision making in robotics," *7th IARP Workshop on Technical Challenges for Dependable Robots in Human Environments*, pp. 15–22, 2010.

[137] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-Aware Motion Planning and Control Using CVaR-Constrained Optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[138] R. Théodose, D. Denis, T. Chateau, V. Frémont, and P. Checchin, "R-AGNO-RPN: A LIDAR-Camera Region Deep Network for Resolution-Agnostic Detection," pp. 1–15, 2020.

[139] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, 2001.

[140] B. Kuipers and Y. T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *Robotics and Autonomous Systems*, vol. 8, no. 1-2, pp. 47–63, 1991.

[141] E. Remolina and B. Kuipers, "Towards a general theory of topological maps," *Artificial Intelligence*, vol. 152, no. 1, pp. 47–104, 2004.

[142] E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey," *Robotics and Autonomous Systems*, vol. 64, pp. 1–20, 2015.

[143] L. Delobel, R. Aufrere, C. Debain, R. Chapuis, and T. Chateau, "A Real-Time Map Refinement Method Using a Multi-Sensor Localization Framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1644–1658, 2018.

[144] C. Aynaud, C. Bernay-Angeletti, R. Aufrère, L. Lequievre, C. Debain, and R. Chapius, "Real-Time Multisensor Vehicle Localization," *IEEE Robotics and Automation Magazine*, no. september, pp. 65–74, 2017.

[145] A. Kasmi, J. Laconte, R. Aufrere, R. Theodose, D. Denis, and R. Chapuis, "An Information Driven Approach for Ego-Lane Detection Using Lidar and OpenStreetMap," in *16th IEEE International Conference on Control, Automation, Robotics and Vision, ICARCV 2020*, 2020, pp. 522–528.

[146] A. Kasmi, J. Laconte, R. Aufrere, D. Denis, and R. Chapuis, "End-to-End Probabilistic Ego-Vehicle Localization Framework," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 146–158, 2021.

[147] DARPA, "DARPA grand challenge 2005 route data definition file," Arlington, VA, USA: Defense Advanced Research Projects Agency, Tech. Rep., 2005.

[148] ——, "Urban challenge: Route network definition file (RNDF) and mission data file (MDF) formats: Technical report," Arlington, VA, USA: Defense Advanced Research Projects Agency, Tech. Rep., 2007.

[149] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[150] M. Haklay and P. Weber, "OpenStreet map: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[151] F. Ramm and J. Topf, *OpenStreetMap: Using and Enhancing the Free Map of the World*. UIT Cambridge Cambridge, 2010, p. 352.

[152] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 420–425, 2014.

[153] F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D. DeWitt, "RoadTracer: Automatic Extraction of Road Networks from Aerial Images Favyen," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4720–4728, 2018.

[154] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler, "Road networks as collections of minimum cost paths," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 108, pp. 128–137, 2015.

[155] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," *European Conference on Computer Vision*, pp. 210–223, 2010.

[156] E. Zoller, "Location Platform Index : Mapping and Navigation. Key vendor rankings and market trends," no. August, pp. 1–24, 2019.

[157] A. Petrovskaya, M. Perrollaz, L. Oliveira, L. Spinello, R. Triebel, A. Makris, J. D. Yoder, C. Laugier, U. Nunes, and P. Bessiere, "Awareness of road scene participants for autonomous driving," *Handbook of Intelligent Vehicles*, vol. 2-2, pp. 1384–1432, 2012.

[158] S. Hwang, N. Kim, Y. Choi, S. Lee, and I. S. Kweon, "Fast multiple objects detection and tracking fusing color camera and 3D LIDAR for intelligent vehicles," *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2016*, pp. 234–239, 2016.

[159] L. Zhang, Q. Li, M. Li, Q. Mao, and A. Nüchter, "Multiple vehicle-like target tracking based on the velodyne lidar," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 8, pp. 126–131, 2013.

[160] F. Fayad and V. Cherfaoui, "Tracking objects using a laser scanner in driving situation based on modeling target shape," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 44–49, 2007.

[161] Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo, and J. Gu, "The obstacle detection and obstacle avoidance algorithm based on 2-D lidar," *2015 IEEE International Conference on Information and Automation, ICIA 2015 - In conjunction with 2015 IEEE International Conference on Automation and Logistics*, pp. 1648–1653, 2015.

[162] A. N. Catapang and M. Ramos, "Obstacle detection using a 2D LIDAR system for an Autonomous Vehicle," *Proceedings - 6th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2016*, no. November, pp. 441–445, 2017.

[163] L. Chen, L. Fan, G. Xie, K. Huang, and A. Nuchter, "Moving-Object Detection from Consecutive Stereo Pairs Using Slanted Plane Smoothing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3093–3102, 2017.

[164] D. Matti, H. K. Ekenel, and J. P. Thiran, "Combining LiDAR space clustering and convolutional neural networks for pedestrian detection," *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2017*, pp. 1–6, 2017.

[165] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[166] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The Stanford Entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[167] M. He, E. Takeuchi, Y. Ninomiya, and S. Kato, "Precise and efficient model-based vehicle tracking method using Rao-Blackwellized and scaling series particle filters," *IEEE International Conference on Intelligent Robots and Systems*, pp. 117–124, 2016.

[168] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *International Journal of Robotics Research*, vol. 29, no. 14, pp. 1707–1725, 2010.

[169] T. D. Vu and O. Aycard, "Laser-based detection and tracking moving objects using data-driven markov chain monte carlo," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3800–3806, 2009.

[170] D. Z. Wang, I. Posner, and P. Newman, "Model-free detection and tracking of dynamic objects with 2D lidar," *International Journal of Robotics Research*, vol. 34, no. 7, pp. 1039–1063, 2015.

[171] W. Xu, J. Snider, J. Wei, and J. M. Dolan, "Context-aware tracking of moving objects for distance keeping," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-Augus, no. Iv, pp. 1380–1385, 2015.

[172] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppe, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe, *et al.*, "Moving Object Detection with Laser Scanners," *Journal of Field Robotics*, vol. 30, no. 1, pp. 17–43, 2013.

[173] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[174] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," 2018.

[175] G. Plastiras, C. Kyrkou, and T. Theocharides, "Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing," *ACM International Conference Proceeding Series*, pp. 779–788, 2018.

[176] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pp. 1–10, 2014.

[177] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An Empirical Evaluation of Deep Learning on Highway Driving," pp. 1–7, 2015.

[178] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[179] F. Mutz, V. Cardoso, T. Teixeira, L. F. Jesus, M. A. Golcalves, R. Guidolini, J. Oliveira, C. Badue, and A. F. De Souza, "Following the leader using a tracking system based on pre-trained deep neural networks," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 4332–4339, 2017.

[180] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 749–765, 2016.

[181] G. K. Kraetzschmar, G. P. Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," *Proceedings of the 5th IFAC/EURON symposium on intelligent autonomous vehicles*, 2004.

[182] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems," in *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010 (ICRA)*, 2010.

[183] A. A. Agha-mohammadi, E. Heiden, K. Hausman, and G. Sukhatme, "Confidence-rich grid mapping," *International Journal of Robotics Research*, pp. 1–23, 2019.

[184] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière, "Bayesian occupancy filtering for multitarget tracking: An automotive application," *International Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, 2006.

[185] C. Chen, C. Tay, C. Laugier, and K. Mekhnacha, "Dynamic environment modeling with gridmap: A multiple-object tracking application," *9th International Conference on Control, Automation, Robotics and Vision, 2006, ICARCV '06*, pp. 1–6, 2006.

[186] T. Gindele, S. Brechtel, J. Schröder, and R. Dillmann, "Bayesian Occupancy Grid Filter for Dynamic Environments Using Prior Map Knowledge," pp. 669–676, 2009.

[187] S. Brechtel, T. Gindele, and R. Dillmann, "Recursive importance sampling for efficient grid-based occupancy filtering in dynamic environments," *Proceedings - IEEE International Conference on Robotics and Automation*, no. June, pp. 3932–3938, 2010.

[188] A. Nègre, L. Rummelhard, and C. Laugier, "Hybrid Sampling Bayesian Occupancy Filter," *Intelligent Vehicles Symposium Proceedings*, pp. 1307–1312, 2014.

[189] L. Rummelhard, A. Negre, and C. Laugier, "Conditional Monte Carlo Dense Occupancy Tracker," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 2485–2490, 2015.

[190] R. Danescu, F. Oniga, and S. Nedevschi, "Particle grid tracking system for stereovision based environment perception," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 987–992, 2010.

[191] ——, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, 2011.

[192] D. Nuss, T. Yuan, G. Krehl, M. Stuebler, S. Reuter, and K. Dietmayer, "Fusion of laser and radar sensor data with a sequential Monte Carlo Bayesian occupancy filter," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-Augus, no. Iv, pp. 1074–1081, 2015.

[193] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss, "Grid-based mapping and Tracking in dynamic environments using a uniform evidential environment representation," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 6090–6095, 2014.

[194] D. Nuss, S. Reuter, M. Thom, T. Yuan, G. Krehl, M. Maile, A. Gern, and K. Dietmayer, "A random finite set approach for dynamic occupancy grid maps with real-time application," *International Journal of Robotics Research*, vol. 37, no. 8, pp. 841–866, 2016.

[195] N. Rexin, D. Nuss, S. Reuter, and K. Dietmayer, "Modeling occluded areas in dynamic grid maps," *International Conference on Information Fusion*, pp. 1–6, 2017.

[196] Á. Llamazares, V. Ivan, E. Molinos, M. Ocaña, and S. Vijayakumar, "Dynamic obstacle avoidance using Bayesian occupancy filter and approximate inference," *Sensors (Switzerland)*, vol. 13, no. 3, pp. 2929–2944, 2013.

[197] M. K. Tay, K. Mekhnacha, C. Chen, M. Yguel, and C. Laugier, "An efficient formulation of the Bayesian occupation filter for target tracking in dynamic environments," *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 1-2, pp. 155–171, 2008.

[198] C. Tay, K. Mekhnacha, M. Yguel, C. Coue, C. Laugier, T. Fraichard, P. Bessiere, C. Tay, K. Mekhnacha, M. Yguel, C. Coue, C. Pradalier, B. Occupation, F. Bessière, and R. P. Reasoning, *The Bayesian Occupation Filter*. Springer, 2008, pp. 77–98.

[199] S. O'callaghan, F. T. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps using gaussian processes," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1054–1060, 2009.

[200] S. T. O'Callaghan, F. T. Ramos, and H. Durrant-Whyte, "Contextual occupancy maps incorporating sensor and location uncertainty," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 3478–3485.

[201] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

[202] ——, "Gaussian process occupancy maps for dynamic environments," *Springer Tracts in Advanced Robotics*, vol. 109, pp. 791–805, 2016.

[203] Y. Yuan, H. Kuang, and S. Schwertfeger, "Fast Gaussian Process Occupancy Maps," *5th International Conference on Control, Automation, Robotics and Vision, ICARCV*, pp. 1502–1507, 2018.

[204] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[205] R. Senanayake, L. Ott, S. O'Callaghan, and F. Ramos, "Spatio-Temporal Hilbert Maps for Continuous Occupancy Representation in Dynamic Environments," *Advances in Neural Information Processing Systems 29*, no. Nips, pp. 3918–3926, 2016.

[206] R. Senanayake and F. Ramos, “Bayesian Hilbert Maps for Continuous Occupancy Mapping in Dynamic Environments,” *Conference on Robot Learning*, vol. 78, no. CoRL, pp. 458–471, 2017.

[207] V. Guizilini, R. Senanayake, and F. Ramos, “Dynamic hilbert maps: Real-time occupancy predictions in changing environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 4091–4097, 2019.

[208] M. Schreiber, V. Belagiannis, C. Glaser, and K. Dietmayer, “Motion Estimation in Occupancy Grid Maps in Stationary Settings Using Recurrent Neural Networks,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 8587–8593, 2020.

[209] S. Hoermann, M. Bach, and K. Dietmayer, “Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2056–2063, 2018.

[210] S. I. Oh and H. B. Kang, “A Modified Sequential Monte Carlo Bayesian Occupancy Filter Using Linear Opinion Pool for Grid Mapping,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 192–198, 2015.

[211] M. Saval-Calvo, L. Medina-Valdés, J. M. Castillo-Secilla, S. Cuenca-Asensi, A. Martínez-Álvarez, and J. Villagrá, “A review of the bayesian occupancy filter,” *Sensors*, vol. 17, no. 2, p. 344, 2017.

[212] M. H. Degroot, “Reaching a consensus,” *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[213] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

[214] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[215] H. Mouhagir, V. Cherfaoui, R. Talj, F. Aioun, and F. Guillemard, “Using evidential occupancy grid for vehicle trajectory planning under uncertainty with tentacles,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1–7, 2018.

[216] O. Takahashi and R. J. Schilling, “Motion Planning in a Plane Using Generalized Voronoi Diagrams,” *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, pp. 143–150, 1989.

[217] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, “PythonRobotics: a Python code collection of robotics algorithms,” pp. 1–8, 2018.

[218] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[219] R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta, "Safe and Reliable Path Planning for the Autonomous Vehicle Verdino," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 2, pp. 22–32, 2016.

[220] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[221] R. Kala and K. Warwick, "Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 72, no. 3-4, pp. 559–590, 2013.

[222] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[223] J. Ziegler, M. Werling, and J. Schröder, "Navigating car-like robots in unstructured environments using an obstacle sensitive cost function," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 787–791, 2008.

[224] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[225] A. Stentz, "Optimal and efficient path planning for partially-known environments," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, no. pt 4, pp. 3310–3317, 1994.

[226] A. Valero-Gomez, J. V. Gomez, S. Garrido, and L. Moreno, "The Path to Efficiency: Fast Marching Method for Safer, More Efficient Mobile Robot Trajectories," *IEEE Robotics and Automation Magazine*, vol. 20, pp. 111–120, 2013.

[227] J. Barraquand and J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[228] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[229] L. Kavraki and J. C. Latombe, "Randomized preprocessing of configuration space for path planning: articulated robots," *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1764–1771, 1994.

[230] N. M. Amato and Y. Wu, "Randomized roadmap method for path and manipulation planning," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, no. April, pp. 113–120, 1996.

[231] P. Svestka and M. H. Overmars, “Motion planning for carlike robots using a probabilistic learning approach,” *International Journal of Robotics Research*, vol. 16, no. 2, pp. 119–143, 1997.

[232] I. Noreen, A. Khan, and Z. Habib, “Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions,” *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, pp. 97–107, 2016.

[233] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 500–505, 1985.

[234] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[235] J. M. Ahuactzin, K. Gupta, and E. Mazer, “Manipulation planning for redundant robots: A practical approach,” *International Journal of Robotics Research*, vol. 17, no. 7, pp. 731–747, 1998.

[236] D. Hsu, J. C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” *International Journal of Computational Geometry and Applications*, vol. 9, no. 4-5, pp. 495–512, 1999.

[237] D. González, J. Pérez, V. Milanés, and F. Nashashibi, “A Review of Motion Planning Techniques for Automated Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[238] M. Brezak and I. Petrovic, “Real-time Approximation of Clothoids With Bounded Error for Path Planning Applications,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 507–515, 2013.

[239] A. Simon and J. C. Becker, “Vehicle guidance for an autonomous vehicle,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 429–434, 1999.

[240] A. Piazzi, C. L. Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, “Quintic G/sup 2/-splines for the iterative steering of vision-based autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 27–36, 2002.

[241] B. Vanholme, S. Glaser, S. Mammar, and D. Gruyer, “Manoeuvre-based trajectory planning for highly autonomous vehicles on real road with traffic,” *2009 European Control Conference, ECC 2009*, vol. 11, no. 3, pp. 3281–3286, 2014.

[242] D. J. Walton, D. S. Meek, and J. M. Ali, “Planar G2 transition curves composed of cubic Bézier spiral segments,” *Journal of Computational and Applied Mathematics*, vol. 157, no. 2, pp. 453–476, 2003.

[243] K. Chu, M. Lee, and M. Sunwoo, “Local path planning for off-road autonomous driving with avoidance of static obstacles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1599–1616, 2012.

[244] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 482–500, 2018.

[245] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," *IEEE International Conference on Robotics and Automation*, vol. 4, pp. 4350–4355, 2004.

[246] T. Wang, L. Zhao, Y. Jia, and J. Wang, "Robot Path Planning Based on Improved Ant Colony Algorithm," *2018 WRC Symposium on Advanced Robotics and Automation, WRC SARA 2018 - Proceeding*, no. February, pp. 115–122, 2018.

[247] C. Blum, "Ant colony optimization: Introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[248] H. Liu, B. Xu, D. Lu, and G. Zhang, "A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 68, pp. 360–376, 2018.

[249] A. Tharwat, M. Elhoseny, A. E. Hassanien, T. Gabel, and A. Kumar, "Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm," *Cluster Computing*, vol. 22, pp. 4745–4766, 2019.

[250] X. Wang, Y. Shi, D. Ding, and X. Gu, "Double global optimum genetic algorithm-particle swarm optimization-based welding robot path planning," *Engineering Optimization*, vol. 48, no. 2, pp. 299–316, 2016.

[251] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Computing*, vol. 21, no. 19, pp. 5829–5839, 2017.

[252] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Applied Soft Computing Journal*, vol. 59, pp. 68–76, 2017.

[253] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. de Paula Veronese, T. Oliveira-Santos, and A. F. De Souza, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113 816, 2021.

[254] M. McNaughton, C. Urmson, J. M. Dolan, and J. W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4889–4895, 2011.

[255] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2061–2067, 2012.

[256] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2015.

[257] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J. W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 250–256, 2015.

[258] D. Fassbender, B. C. Heinrich, and H. J. Wuensche, "Motion planning for autonomous vehicles in highly constrained urban environments," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 4708–4713, 2016.

[259] S. Joachim, G. Tobias, J. Daniel, and D. Riidiger, "Path planning for cognitive vehicles using risk maps," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 1119–1124, 2008.

[260] R. R. Radaelli, C. Badue, M. A. Gonçalves, T. Oliveira-Santos, and A. F. De Souza, "A motion planner for car-like robots based on rapidly-exploring random trees," *Ibero-American conference on artificial intelligence*, pp. 469–480, 2014.

[261] M. Du, T. Mei, H. Liang, J. Chen, R. Huang, and P. Zhao, "Drivers' visual behavior-guided RRT motion planner for autonomous on-road driving," *Sensors (Switzerland)*, vol. 16, no. 1, pp. 1–19, 2016.

[262] K. Yang, S. Keat Gan, and S. Sukkarieh, "A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV," *Advanced Robotics*, vol. 27, no. 6, pp. 431–443, 2013.

[263] C. Alia, T. Gilles, T. Reine, and C. Ali, "Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-Augus, no. Iv, pp. 674–679, 2015.

[264] H. Mouhagir, R. Talj, V. Cherfaoui, F. Aioun, and F. Guillemard, "Integrating safety distances with trajectory planning by modifying the occupancy grid for autonomous vehicle navigation," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1114–1119, 2016.

[265] M. Himmelsbach, T. Luettel, F. Hecker, F. von Hundelshausen, and H. J. Wuensche, "Autonomous Off-Road Navigation for MuCAR-3: Improving the Tentacles Approach: Integral Structures for Sensing and Motion," *KI - Kunstliche Intelligenz*, vol. 25, no. 2, pp. 145–149, 2011.

[266] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - A local, continuous method," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 450–457, 2014.

[267] E. Heiden, K. Hausman, G. S. Sukhatme, and A. A. Agha-Mohammadi, "Planning high-speed safe trajectories in confidence-rich maps," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 2880–2886, 2017.

[268] J. Vaščák, “Navigation of mobile robots using potential fields and computational intelligence means,” *Acta Polytechnica Hungarica*, vol. 4, no. 1, pp. 63–74, 2007.

[269] T. M. Howard and A. Kelly, “Optimal rough terrain trajectory generation for wheeled mobile robots,” *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[270] D. Ferguson, T. M. Howard, and M. Likhachev, “Motion planning in urban environments,” *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[271] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, “Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles,” *Mechanical Systems and Signal Processing*, vol. 87, pp. 118–137, 2017.

[272] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese, and A. F. De Souza, “A Model-Predictive Motion Planner for the IARA autonomous car,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 225–230, 2017.

[273] C. Fulgenzi, A. Spalanzani, and C. Laugier, “Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid,” *IEEE International Conference on Robotics and Automation*, pp. 1610–1616, 2007.

[274] D. Lachapelle, T. Humphreys, L. Narula, P. Iannucci, and E. Moradi-Pari, “Automotive Collision Risk Estimation under Cooperative Sensing,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 9200–9204, 2020.

[275] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi, “STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Off-road Navigation,” 2021.

[276] A. Slavík and Univerzita Karlova. Matematicko-fyzikální fakulta., *Product Integration, Its History and Applications*. Matfyzpress Prague, 2007, vol. 1.

[277] S. Rohou, L. Jaulin, L. Mihaylova, F. L. Bars, S. Veres, S. Rohou, L. Jaulin, L. Mihaylova, F. L. Bars, S. V. Reliable, S. Rohou, L. Jaulin, L. Mihaylova, and F. Le, “Reliable non-linear state estimation involving time uncertainties,” *Automatica*, vol. 93, pp. 379–388, 2018.

[278] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[279] R. Senanayake and F. Ramos, “Directional Grid Maps: Modeling Multimodal Angular Uncertainty in Dynamic Environments,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 3241–3248, 2018.

[280] B. Gerkey and K. Konolige, “Planning and control in unstructured terrain,” *ICRA Workshop on Path Planning on Costmaps*, 2008.

[281] L. Rummelhard, A. Nègre, M. Perrollaz, and C. Laugier, “Probabilistic Grid-based Collision Risk Prediction for Driving Application,” *Experimental Robotics*, pp. 821–834, 2014.

[282] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, “Long-term 3D map maintenance in dynamic environments,” *Proceedings - IEEE International Conference on Robotics and Automation*, no. 1, pp. 3712–3719, 2014.

[283] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” no. CoRL, pp. 1–16, 2017.

[284] J. Laconte, A. Kasmi, F. Pomerleau, R. Chapuis, L. Malaterre, C. Debain, and R. Aufrère, “A Novel Occupancy Mapping Framework for Risk-Aware Path Planning in Unstructured Environments,” *Sensors*, vol. 21, no. 22, p. 7562, 2021.

[285] J. Morceaux, J. Laconte, E. Randriamiarintsoa, T. Morell, L. Malaterre, D. Denis, R. Aufrere, and R. Chapuis, “Toward a Generalized Risk Assessment Method on Occupancy Grids,” *Late Breaking Results Posters - 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[286] J. Laconte, S. P. Deschênes, M. Labussière, and F. Pomerleau, “Lidar measurement bias estimation via return waveform modelling in a context of 3D mapping,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, 2019, pp. 8100–8106.

[287] J. Laconte, A. Kasmi, F. Pomerleau, R. Chapuis, L. Malaterre, C. Debain, and R. Aufrère, “A Survey of Localization Methods for Autonomous Vehicles in Highway Scenarios,” *Sensors*, 2021.

[288] M. Vaidis, J. Laconte, V. Kubelka, and F. Pomerleau, “Improving the Iterative Closest Point Algorithm using Lie Algebra,” *IROS 2020 Workshop: Bringing geometric methods to robot learning, optimization and control*, 2020.

[289] D. Baril, S.-P. Deschênes, O. Gamache, M. Vaidis, D. LaRocque, J. Laconte, V. Kubelka, P. Giguère, and F. Pomerleau, “Kilometer-scale autonomous navigation in subarctic forests: Challenges and lessons learned,” *Submitted to Field Robotics (FR)*, 2021.