Stratégie de perception active pour l'interprétation de scènes

Coralie Bernay-Angeletti

▶ **To cite this version:**

Coralie Bernay-Angeletti. Stratégie de perception active pour l'interprétation de scènes. Autre. Université Blaise Pascal - Clermont-Ferrand II, 2016. Français. NNT : 2016CLF22710 . tel-01420115

HAL Id: tel-01420115
https://theses.hal.science/tel-01420115

Submitted on 20 Dec 2016

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.

N° d'ordre : D.U : 2710
EDSPIC: 763

Université Blaise Pascal - Clermont 2

École doctorale
Sciences pour l'Ingénieur de Clermont-Ferrand

THÈSE
présentée par

Coralie BERNAY-ANGELETTI

Pour obtenir le grade de
Docteur d'Université
Spécialité : "Vision pour la robotique"

Stratégie de Perception Active pour l'interprétation de scène

---

Préparée à l'Institut Pascal
Soutenue publiquement le 29 juin 2016

Jury		
François CHARPILLET	Directeur de Recherche (Inria, Univ. Lorraine)	Président de jury
Helder ARAÙJO	Professeur (Polo II, Univ. Coimbra, Portugal)	Rapporteur
Samia BOUCHAFA	Professeur (IBISC, Univ. Evry Val d'Essonne)	Rapporteur
Romuald AUFRÈRE	Maître de Conférences (Univ. Blaise Pascal)	Examinateur
Christophe DEBAIN	Chargé de Recherches (Irstea Clermont)	Examinateur
Roland CHAPUIS	Professeur (Univ. Blaise Pascal)	Directeur de thèse

Ce travail a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre du programme Investissements d'avenir dans le cadre des projets EquipEx Robotex (ANR-10-EQPX-44) et LabEx IMobS3 (ANR-10-LABX-16-01), d'une aide de l'Union Européenne au titre du Programme Compétitivité Régionale et Emploi 2007-2013 (FEDER – Région Auvergne), d'une aide de la Région Auvergne et de l'Institut Français de Mécanique Avancée.

Remerciements

Je tiens tout d'abord à remercier mes encadrants : Roland Chapuis et Romuald Aufrère pour m'avoir guidée tout au long de cette thèse. Sans leurs conseils, leurs remarques et leurs relectures, nul doute que ce manuscrit, aboutissement du long cheminement qu'est la thèse, n'aurait pas pu voir le jour.

Je remercie mes rapporteurs : Madame Samia Bouchafa et Monsieur Helder Araujo pour l'attention apportée à mon travail de thèse. Je remercie l'ensemble du jury et en particulier le président du jury, Monsieur François Charpillet pour leur sollicitude durant ma soutenance de thèse. Je remercie également tous les membres du jury pour leur intérêt pour mon travail de thèse et leurs questions.

Je remercie particulièrement Claude Aynaud pour nos nombreuses discussions passionnées et passionnantes autour des probabilités (et de leurs calculs) puis des réseaux bayésiens. Je me rappelle particulièrement de débats endiablés pour finalement s'apercevoir que nous étions d'accord depuis le début et que l'incompréhension venait juste d'une mauvaise modélisation (on ne vantera jamais assez les mérites d'une modélisation simple à comprendre).

Je remercie Jean-Charles Quinton pour avoir enrichi mes connaissances sur la perception humaine, les études permettant de mieux comprendre le fonctionnement humain et les exemples d'imitations algorithmiques de tels comportement.

Je remercie toutes les personnes que je n'ai pas encore citées qui m'ont aidée dans mes travaux que ce soit par leurs remarques, par leur aide technique ou simplement par leur bonne humeur dont Clément Deymier, Ange Nizard, Thanh Tin Nguyen, Guillaume Bresson, Thomas Féraud, Datta Ramadasan, Laetitia Lamard, Mihai Chirca, Laurent Lequièvre, Laurent Delobel, Laurent Malaterre, Laurent Trassoudaine, Serge Alizon, Céline Teulière et bien d'autres encore.

Je remercie mes parents et mes soeurs pour avoir parcouru de longs kilomètres pour venir assister à ma soutenance et avoir égayé les derniers jours de préparation avant la soutenance. Je remercie également mon frère qui aurait bien voulu participer lui aussi même s'il n'a pas pu. Je remercie ma fille Ambre pour m'avoir inspiré. Je remercie mon fils Hector pour avoir gentiment attendu avant de naître. Enfin, je tiens à remercier mon mari Benoit pour sa patience infinie et sa vision toujours positive.

Résumé

La perception est le moyen par lequel nous connaissons le monde extérieur. C'est grâce à nos perceptions que nous sommes capables d'interagir avec notre environnement et d'accomplir de nombreuses actions du quotidien comme se repérer, se déplacer, reconnaître des objets et autres. Cette perception n'est pas juste passive comme peut l'être une sensation, elle comporte des aspects actifs. En particulier, elle peut être orientée dans un but précis, permettant de filtrer les données pour ne traiter que les plus pertinentes.

Si la perception humaine est particulièrement efficace, la perception artificielle, elle, demeure un problème complexe qui se heurte à de nombreuses difficultés. Ainsi, les changements de conditions de perception comme des modifications de l'illumination ou des occultations partielles de l'objet à percevoir doivent pouvoir être gérées efficacement. Pour résoudre ces difficultés, s'inspirer de la perception humaine semble être une piste intéressante.

Ce manuscrit propose un système de perception polyvalent et générique reposant sur une stratégie de perception active. Pour ce faire, nous proposons un algorithme Top-Down utilisant un modèle en parties. Le problème de perception est transformé en un problème d'estimation d'un vecteur de caractéristiques. La détection des différentes parties permet de réaliser cette estimation.

Le système de perception proposé est un algorithme itératif multi-capteurs. À chaque itération, il sélectionne au mieux, en fonction des objectifs fixés par l'application, la partie à détecter ainsi que les meilleurs capteur et détecteur compatibles. Un réseau bayésien est utilisé pour prendre en compte les événements incertains pouvant survenir lors de ce processus comme la défaillance d'un détecteur ou la non existence potentielle d'une partie donnée. Un processus de focalisation à la fois spatiale et de caractéristiques permet d'améliorer la détection en augmentant le rapport signal sur bruit, en restreignant la zone de recherche pour une partie et en éliminant certains des candidats trouvés. Ce processus de focalisation permet aussi de réduire les temps de calcul et de restreindre l'influence des distracteurs.

L'ajout de nouveaux capteurs, détecteurs ou parties se fait simplement. De plus, l'utilisation d'un réseau bayésien permet une grande flexibilité au niveau de la modélisation des événements pris en compte : il est facile de rajouter de nouveaux événements pour obtenir une modélisation plus réaliste.

L'algorithme proposé a été utilisé pour plusieurs applications incluant de la reconnaissance d'objets, de l'estimation fine de pose et de la localisation.

Mots clefs : approche Top-Down, réseau bayésien, focalisation, modèle en parties, multi-capteurs

Abstract

Perception is the way by which we know the outside world. Thanks to our perceptions we are able to interact with our environment and to achieve various everyday life actions as locating or moving in an environment, or recognizing objects. Perception is not passive whereas sensations are, it has active components. In particular, perception can be oriented for a specific purpose allowing to filter data and to take care only of the most relevant.

If human perception is particularly effective, artificial perception remains a complex problem with a lot of non solved difficulties. For example, changes of perception conditions as modification of illumination or partial occultation of the searched object must be effectively managed.

This thesis proposes a system of perception based on a strategy of active perception which can adapt itself to various applications. To do it, we propose an algorithm Top-Down using a part-based model. The problem of perception is transformed into a problem of estimation of a characteristics vector. The detection of the different parts constituting the searched object allows to realize this estimation.

The proposed perceptive system is an iterative and multi-sensors algorithm. In every iteration, it selects, at best, according to the application objectives, the part to detect and the best compatible sensor and detector. A bayesian network is used to take into account uncertain events which can arise during this process as detector failure or potential non existing part. A focus process consisting of a spatial focus and of a characteristics focus, improves the detection by restricting the search area, by improving the signal to noise ratio and by eliminating some erroneous candidates. This focus process also allows to reduce computation time and to restrict influence of distractors.

Adding a part, a sensor or a detector is simple. Furthermore, the use of a bayesian network allows to be flexible in the events modelisation : it is easy to add new events to obtain a more realistic modelisation.

The proposed algorithm has been used for several applications including object's recognition, fine pose estimation and localization. So, it is multi-purpose and generic.

Keywords : Top-Down approach, Bayesian Network, focusing, part based models, multi-sensors

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

Il peut sembler assez aisé de donner une définition de ce qu'est la perception. Ainsi, il est commun de dire qu'elle est le moyen par lequel nous connaissons le monde extérieur. Néanmoins, la perception ne saurait se résumer à un phénomène passif identifiable à une sensation. Au contraire, elle est active et nécessite une certaine forme d'organisation, de structuration et de sélection des données pertinentes. Les informations en provenance du monde extérieur sont sélectionnées, décodées, interprétées. La perception est une lecture de la réalité. Cette lecture passe par trois étapes au moins : une étape sensorielle, une étape d'analyse et une étape d'interprétation.

La première étape consiste en la réception des stimuli extérieurs via les organes sensoriels. Ceux-ci sont propres à chaque espèce et définissent les limites de la perception. Par exemple, la vision humaine est limitée à un spectre précis : ni les ultraviolets ni la lumière infrarouge n'en font partie, ce qui n'est pas le cas chez certains insectes. De même la gamme des perceptions sonores est limitée : un être humain ne peut pas entendre les ultrasons ni les infrasons alors qu'ils existent. Le rapport à l'environnement extérieur est donc modelé par la sensibilité des capteurs sensoriels.

Dès lors survient une étape d'analyse. Elle consiste à dépasser les strictes données sensorielles pour les mettre en forme. Prenons l'exemple d'étoiles dans le ciel, les capteurs sensoriels sont sensibles simplement à la lumière émise par ces étoiles mais bien qu'elles soient dispersées dans le ciel, sans ordre apparent, le cerveau a tendance à regrouper spontanément les étoiles entre elles selon une loi de proximité. Ainsi sont formées les constellations. Une autre loi de la perception veut que les formes géométriques simples comme les lignes, les cercles, les carrés ou les rectangles seront immédiatement détectées. Les formes ainsi reconnues aident à organiser les données de l'environnement en repérant les distinctions du fond, de la la forme, des contours des objets et en déformant ou en complétant au besoin les éléments manquants pour redonner une certaine cohérence.

La troisième étape est celle de l'interprétation des données. Elle consiste à attribuer une signification à l'information. Cette signification sera dépendante des représentations d'une époque, des modèles culturels mais également des connaissances et des acquis de la personne. Ainsi, en regardant le ciel, le néophyte et l'astronome ne verront pas les mêmes constellations. La perception du monde est donc dépendante des capacités de nos organes sensoriels mais aussi de nos connaissances et de nos objectifs.

Ce système de perception est indispensable pour interagir efficacement avec l'environnement. Si les êtres humains sont naturellement dotés d'un tel système, il n'en va pas de même pour les systèmes artificiels. Pourtant, avec l'amélioration des capteurs et la

progression des ordinateurs, il nous parait judicieux de se pencher sur la problématique d'un système de perception artificielle.

L'objectif général de cette thèse est donc de proposer un système de perception reposant sur une stratégie de perception active pouvant s'adapter à différentes applications comme la reconnaissance d'objets ou la localisation de robot mobile.

L'algorithme proposé pour réaliser ce système de perception devra donc être générique afin de s'adapter à différentes situations. Il devra permettre la gestion de plusieurs capteurs sans avoir besoin de modifier son fonctionnement. Il devra fusionner de manière cohérente les données en provenance de ces différents capteurs. À chaque instant, les connaissances disponibles devront pouvoir être prises en compte qu'il s'agisse de connaissances a priori (un piéton marche sur le sol et ne vole pas) ou de connaissances acquises (la personne a les yeux verts car un œil vert a été détecté). Les actions de perception à effectuer dépendront de l'objectif visé. Cet objectif sera en fait double : il contiendra un objectif de précision, qui définira ce qu'il faut percevoir et avec quel niveau de détail, ainsi qu'un objectif de confiance, qui définira la certitude que nous voulons avoir sur la reconnaissance.

Les principales contributions de cette thèse sont :

— **Proposition de modélisation générique**
Le problème de perception qu'il s'agisse, par exemple, d'un problème de reconnaissance d'objets [1, 2] ou d'un problème de localisation [5, 6] est transformé en un problème d'estimation d'un vecteur de caractéristiques.

— **Mise en place d'une stratégie de sélection**
Parmi l'ensemble des entités constituées d'une partie, d'un capteur et d'un détecteur, l'algorithme sélectionnera, en fonction des objectifs et de la situation courante, celle qu'il estimera être la meilleure [1].

— **Construction d'un réseau bayésien adapté**
Afin de gérer les événements incertains qui peuvent survenir au cours du processus de perception, comme un échec de détection lié à une erreur du détecteur ou à la non visibilité d'une partie, nous proposons l'utilisation d'un réseau bayésien [2]. Ce dernier permet d'avoir une représentation visuelle des différents événements pris en compte et des liens entre ces événements. De plus, les réseaux bayésiens sont flexibles et peuvent être complexifiés au fur et à mesure des besoins afin de s'approcher d'une modélisation réaliste.

— **Proposition d'un processus de focalisation**
Nous proposons un système de focalisation [3] qui comporte deux facettes : une focalisation spatiale et une focalisation de caractéristiques. La première consiste à éviter de traiter toute l'image capteur et à ne considérer qu'une zone d'intérêt recalculée au fur et à mesure pour chaque partie. La deuxième permet de ne sélectionner dans cette zone d'intérêt que les données pertinentes en fonction des connaissances sur l'objet.

— **Mise en place d'un système de remise en cause**
Nous pouvons au cours du processus de reconnaissance avoir effectué une mauvaise association. Lorsqu'il y a plusieurs candidats pour une partie, le candidat choisi n'est potentiellement pas le bon. Nous souhaitons être capable, lorsque cela s'avère nécessaire de revenir sur une telle association et de modifier le choix jusqu'à la bonne attribution. Nous proposons donc un système permettant d'estimer quand un tel

retour est nécessaire et capable de l'effectuer [2].

— **Démonstration de la généricité et de la polyvalence de notre algorithme**
Notre algorithme est capable de traiter différents problèmes. Nous proposons de montrer sa généricité en l'utilisant pour des applications variées incluant de la reconnaissance d'objets [1, 2, 4], de l'estimation fine de pose [3] et de la localisation de véhicule [5, 6].

L'élaboration, la conception et la mise en application de notre algorithme de perception seront présentées à travers cinq chapitres. Dans le chapitre I, nous aborderons la perception humaine et un état de l'art sur les méthodes s'inspirant de celle-ci. Ensuite, dans le chapitre II, nous expliquerons les objectifs visés, les choix stratégiques faits en conséquence ainsi que les grandes lignes de notre algorithme. Le chapitre III détaillera les différentes étapes de notre approche en expliquant les éléments pris en compte et les hypothèses de modélisation faites. Dans le chapitre IV nous verrons quelques exemples d'utilisation de notre algorithme. Enfin, le chapitre V présentera la conclusion de notre travail et différentes perspectives en vue de l'améliorer.

Chapitre I

Perception Visuelle

« *La vision est l'art de voir les choses invisibles.* »

Jonathan Swift

Sommaire

I.1 Introduction

L'objectif général de cette thèse est de proposer un système de perception reposant sur une stratégie de perception active pouvant s'adapter très facilement à différentes applications de perception artificielle. Si nous revenons sur chacun de ces termes, selon le dictionnaire Larousse, la **perception** peut avoir les sens suivants :

— action de percevoir par les organes des sens

— idée, compréhension plus ou moins nette de quelque chose

— événement cognitif dans lequel un stimulus ou un objet présent dans l'environnement immédiat d'un individu lui est représenté dans son activité psychologique interne en principe de façon consciente : fonction psychologique qui assure ces perceptions

La **stratégie**, quant à elle, est définie comme l'art de coordonner des actions, de manœuvrer habilement pour atteindre un but. Toujours selon le même dictionnaire, **actif** peut signifier qui agit avec efficacité, qui donne des résultats ; efficace. L'objectif est donc de mettre en œuvre un ensemble d'actions coordonnées afin de parvenir avantageusement à un objectif de perception.

La perception comporte donc plusieurs facettes : d'une part elle implique d'avoir des *organes de sens* permettant la récolte d'informations sur le monde extérieur. Si les *organes des sens* sont de manière évidente de réels organes pour un être humain ou un animal, il est possible de leur accorder un sens plus large afin de pouvoir inclure la perception artificielle par des robots. Dans ce cas, les *organes des sens* peuvent être assimilés à des capteurs : les yeux qui confèrent la vue sont remplacés par des caméras, le repérage à l'oreille peut se faire via des télémètres ou des LIDARS etc. Dans le vivant, tous les êtres n'ont pas la même vue (la vue d'une mouche est assez différente de celle d'un humain). De façon similaire, différents types de capteurs correspondant pourtant à un même sens (par exemple une caméra omnidirectionnelle et une caméra standard) amèneront différentes perceptions. D'autre part, la perception met en place des processus cognitifs (dont l'équivalent artificiel serait des algorithmes) permettant de transformer les sensations immédiates en une représentation ou une compréhension de quelque chose.

En effet, c'est grâce à nos perceptions que nous pouvons effectuer diverses actions du quotidien comme se repérer et se déplacer dans notre environnement, savoir éviter les obstacles ou encore reconnaître des objets, des personnes, des mots et interagir en conséquence. Souvent, les perceptions sensorielles se combinent afin de conforter à un instant donné notre représentation des événements : ainsi la tristesse ou la joie d'une personne pourra être identifiée à la fois grâce à des informations visuelles (elle pleure ou elle sourit) mais aussi des informations sonores (un ton déprimé ou au contraire joyeux). De la même manière, la perception artificielle pourra intégrer plusieurs capteurs afin d'en recouper les informations.

Pour illustrer nos propos, nous prendrons souvent comme exemple la perception pour la reconnaissance d'objets du fait de l'ampleur des travaux que cela a suscité. Néanmoins, il ne faut pas perdre de vue que la finalité de notre algorithme sera plutôt orientée vers la manière de percevoir l'information (éventuellement issue de plusieurs capteurs) pour atteindre un objectif donné (reconnaissance d'objets mais aussi localisation d'un robot ou encore l'analyse de scènes, etc).

La perception artificielle est un problème complexe [Prasad, 2012] qui se heurte à de nombreuses difficultés. Parmi ces dernières, certaines sont inhérentes à la nature même de la tâche. Par exemple dans le cas de la reconnaissance d'objets, pour être robustes et fonctionnels, les algorithmes doivent être capables de détecter l'objet désiré dans des conditions variables. L'objet doit être reconnu indépendamment (le plus possible) de son environnement : il faut donc faire attention à l'influence potentielle de l'illumination (pour des caméras standards) ou encore des éventuels changements de température de l'objet (pour des caméras infrarouges) ou bien de perturbations électromagnétiques (pour des magnétomètres par exemple). L'orientation et la position de l'objet dans l'image capteur doivent également être sans conséquence ou presque pour la reconnaissance. Or un objet perçu de face, de profil ou de dos peut avoir un aspect très différent. Enfin, il existe le problème de l'occultation : il est possible (et même probable selon les objets et les lieux) que dans un environnement non maîtrisé, une partie ou la totalité de l'objet ne soit pas visible car cachée (ou étouffée dans le cas de son) par un autre objet. La figure I.1 montre quelques illustrations de ces problèmes pour le cas particulier des images.

(a) Photo de jour (b) Photo de nuit (c) Photo de dos

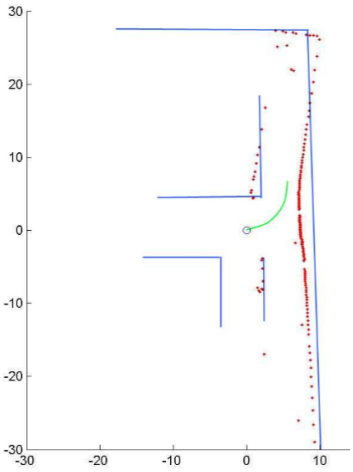
FIGURE I.1 – Statue de Vercingétorix à Clermont-Ferrand

D'autres difficultés proviennent quant à elles des capteurs utilisés pour obtenir les données sur lesquelles seront exécutés les algorithmes de perception. Qu'il s'agisse de caméras, de lidars, d'odomètres ou autres, les données capteurs ne sont pas parfaites et présentent différents défauts augmentant la complexité de la tâche : présence de bruits, de flou en particulier en cas de mouvement (du capteur ou de la cible) ou encore de déformations. La figure I.2 donne quelques exemples de ces problèmes dans des cas d'interprétations de scène ou de reconnaissance d'objets avec des données fournies par une caméra ou un LIDAR. Pour les données LIDAR, du fait du mouvement du robot (en vert), les données (rouges) sont distordues par rapport à la réalité (murs représentés en bleu).

Enfin, une dernière catégorie de difficultés est issue de la définition même de la tâche de perception. Si percevoir c'est recevoir de l'information par les organes des sens, c'est également permettre la compréhension et l'interprétation des données sensorielles. Or plusieurs éléments peuvent avoir une interprétation similaire alors que les données associées (apparence, toucher ou autre) seront assez dissemblables. À l'inverse, des éléments avec des données associées similaires n'auront pas nécessairement la même signification ce qui peut s'avérer problématique. C'est l'exemple de la classification dans le cas particulier de la reconnaissance d'objets : une catégorie (classe) d'objets assez large posera le problème de la variabilité des objets appartenant à cette classe. Ainsi, dans la classe "téléphone", un téléphone portable actuel et un téléphone fixe des années vingt se ressembleront assez peu. À l'inverse, un vélo et une moto pourront potentiellement être confondus. La figure I.3

(a) Exemple de distorsion (caméra)

(b) Exemple de flou (caméra)



(c) Exemple de distorsion (LIDAR) due au mouvement du véhicule [Vivet, 2011]

FIGURE I.2 – Exemples d'imperfections liées au capteur

permet d'illustrer ces problèmes pour un capteur orienté vision (caméra par exemple) à l'aide des catégories "voitures" et "camionnette". Cette difficulté d'interprétation n'est évidemment pas spécifique aux problèmes de vision : différentes sirènes seront interprétées comme un signal d'alarme malgré des sonorités fort peu semblables tandis que des sons proches (comme des pleurs d'enfants) pourront être différenciés.

Pourtant, la perception est un enjeu important. En effet, elle répond à de multiples besoins dans des domaines variés. Percevoir son environnement est nécessaire pour interagir avec lui. La perception peut être utile aussi bien pour des problèmes de localisation que pour des problèmes d'interprétation ou de reconnaissance d'objets. Pour illustrer la diversité des applications, nous allons dans les paragraphes suivants étudier plus en détails quelques exemples pour la reconnaissance d'objets même si la perception ne saurait être restreinte à ce seul champ.

La perception pour la reconnaissance d'objets peut permettre d'automatiser des tâches auparavant longues, fastidieuses et/ou répétitives dont la monotonie rend susceptible les erreurs quand ces tâches sont effectuées par des personnes. Par exemple, le contrôle de la conformité de pièces [Viana et al., 2013] lors d'usinages industriels se prête bien à une automatisation du processus. Les conditions pouvant être maîtrisées (position du capteur connu, réglage de l'éclairage, etc), la tâche de reconnaissance en est facilitée et peut alors se concentrer pour avoir des mesures précises.

À l'inverse, le dénombrement automatique d'une population (animale [Descamps et al., 2011], végétale, humaine ou même d'objets tels que des voitures) dans une série d'images demande des algorithmes traitant des environnements plus variables mais dont l'objectif n'est pas une pose précise mais simplement le nombre d'objets qui sont présents dans la scène. Ce dénombrement permet de réaliser des statistiques afin d'estimer, par exemple, la fréquentation d'un lieu selon les heures de la journée (comme pour un trafic routier) ou selon les saisons (comme pour la migration des oiseaux).

(a) Image de Ferrari

(b) Image de Nano

(c) Image de Coccinelle

(d) Image de Logan (voiture)

(e) Image d'une camionnette

FIGURE I.3 – Exemple de problèmes liés aux catégories d'objets pour les catégories "voiture" et "camionnette". Première ligne : exemples de voitures très dissemblables. Deuxième ligne : similarité entre un break et une camionnette.

Un dernier type d'applications dont l'automatisation est grandement appréciable est l'indexation automatique d'images. En effet, grâce à Internet, à la disponibilité des capteurs tels que les appareils photos et à l'accessibilité des moyens de stockages dont les clés usb ou les disques durs, les bases de données d'images ont grandement augmenté, créant le besoin de pouvoir accéder aux images pertinentes selon les besoins ; par exemple, la recherche d'images correspondant à des mots clés comme "chiens" ou "chats".

En dehors de l'automatisation totale d'une tâche, la reconnaissance d'objets a d'autres atouts dans le monde du travail. Elle peut être une aide à un opérateur humain. Entre autres, une application est la vidéosurveillance [Valera and Velastin, 2005]. La reconnaissance d'objets sert alors à détecter des anomalies comme la présence d'une personne dans une zone interdite au public (l'opérateur humain pouvant alors confirmer ou infirmer le droit potentiel de la personne à s'y trouver) ou encore l'abandon d'un objet suspect dans un lieu à risques (abandon d'un sac à dos dans un aéroport ou dans une gare). La personne humaine regarde alors l'alerte donnée par l'algorithme et juge en fonction de la situation des suites à donner.

La détection d'anomalies commence aussi à apparaître dans le domaine médical pour l'aide au diagnostic : il s'agit de mettre en exergue dans des images médicales (issues de scanners par exemple) des zones paraissant suspectes ou de proposer directement un premier diagnostic [Ramírez et al., 2013].

Récemment également, la réalité augmentée se fraye un chemin en dehors des jeux vidéos et la reconnaissance d'objets est le prérequis nécessaire afin que les objets virtuels insérés dans l'image soient bien positionnés.

Il peut s'agir d'une aide interactive pour comprendre le fonctionnement d'une machine [Henderson and Feiner, 2009] : des flèches montrent la partie de l'objet sur laquelle appuyer ou qu'il faut tourner pour déclencher l'action suivante. Reconnaître et localiser précisément l'objet dans l'image est nécessaire pour que les flèches aient un sens compréhensible.

Une autre application est liée à l'aspect commercial [Parhizkar et al., 2011] : permettre à un éventuel client de voir la voiture (ou un autre objet en présentation) dans d'autres coloris que celui en exposition. Essayer "virtuellement" un produit résout certains problèmes comme la difficulté d'essayer une monture sans verre avec une vue faible (sans lunettes difficile d'y voir et la superposition ne donne qu'une idée partielle) ou encore pour les commandes par internet de maquillages, vernis ou autres dont avoir une vision sur soi est un facteur de décision. Comme précédemment, une bonne reconnaissance d'objets (le visage pour les applications de maquillage par exemple) est indispensable pour bien positionner les éléments virtuels et apporter une plus value.

L'ensemble des applications citées ci dessus ne prétend pas être exhaustif néanmoins il montre bien l'intérêt croissant de développer des approches de perception artificielle en particulier pour reconnaître des objets dans une scène. Il est donc d'autant plus essentiel de résoudre les difficultés liées à cette dernière afin d'améliorer sa robustesse ou d'élargir encore ses applications potentielles.

Un point particulièrement intéressant est de remarquer que chez l'être humain la tâche de reconnaissance est une tâche aisée du quotidien [Tarr, 2000]. Or les êtres humains ne sont pas passifs dans la tâche de perception [Bajcsy, 1988, Pezzulo, 2008]. Comme l'écrit Craik [Craik, 1967] : *If the organism carries a "small-scale model" of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer and more competent manner to the emergencies which face it.* Les humains prennent donc en compte leurs connaissances précédentes (innées ou acquises) et leurs objectifs [Gibson, 1962] afin de sélectionner les informations pertinentes pour comprendre et analyser la situation. Ils ne sont pas complètement passifs au monde extérieur mais agissent pour accéder à l'information dont ils ont besoin [Ballard and Brown, 1992, Prescott et al., 2011]. Grâce à cela, ils sont capables de prendre des décisions comme trancher dans la catégorisation d'une image (est-ce un chat ou un chien ? [Quinton et al., 2014]).

I.2 Perception Visuelle Naturelle

Nous avons vu dans la partie précédente que la perception artificielle en toute circonstance est délicate malgré les nombreux intérêts qu'un système de perception automatique peut avoir. D'un autre côté, la perception humaine est particulièrement efficiente. L'objectif de ce travail de thèse étant de développer une stratégie générique pour la perception, étudier le système de perception humain semble donc judicieux afin de s'en inspirer. Nous allons nous pencher plus particulièrement sur la perception visuelle qui nous semble être assez représentative de notre propos tout en ayant conscience qu'il ne s'agit que d'une partie de la perception naturelle humaine.

I.2.1 Eléments de physiologie du système humain visuel

Les yeux sont les capteurs du système visuel humain. Ils transmettent l'information visuelle qui sera traitée par différents composants du cerveau. Or l'œil dispose de deux zones distinctes de vision : la vision fovéale et la vision périphérique. Lorsqu'un stimulus

est perçu par le centre de la rétine appelée fovéa, le système visuel est particulièrement sensible au détail de ce stimulus qu'il code tout au long de la chaîne du traitement de l'information visuelle. En revanche, lorsque ce même stimulus est perçu dans la périphérie rétinienne, une partie de l'information plus ou moins importante pour la reconnaissance de ce stimulus est perdue. En effet, l'acuité visuelle ou résolution spatiale, est maximale dans la fovéa et se dégrade progressivement dans la périphérie [Legrand, 1972, Yssaad, 2001].

La figure I.4 illustre ce principe : en fixant la lettre 'π', celle-ci est identifiable bien que cela soit difficile. Par contre, en fixant le point à gauche, la reconnaissance du 'π' dans la partie périphérique droite s'avère très difficile voire impossible. Un processus d'attention sera donc nécessaire pour se concentrer sur une partie spécifique de la scène. Des différences anatomiques et physiologiques de l'être humain expliquent ce fonctionnement variable selon l'orientation du regard.

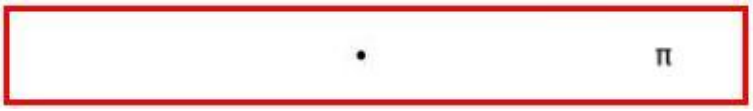


FIGURE I.4 – Exemple illustrant la différence entre la vision fovéale et la vision périphérique.

La rétine dispose de deux types de photorécepteurs inégalement répartis : les cônes et les bâtonnets [Gregory et al., 2000]. Les cônes assurent la vision dans le domaine photopique (vision diurne) et traitent le détail et la couleur. Dans l'œil humain, il y a généralement 3 types de cônes ayant une sensibilité plus grande à certaines radiations de longueurs d'onde comprises entre 400 et 700 nm (voir figure I.5) : les cônes bleus sensibles aux radiations de basses longueurs d'onde ou cônes cyanolabes (420 nm), les cônes verts sensibles aux radiations de moyennes longueurs d'onde ou cônes chlorolabes (534 nm), et les cônes rouges sensibles aux radiations de grandes longueurs d'onde ou cônes érytholabes (564 nm) (ces derniers réagissant d'ailleurs principalement aux radiations provoquant la sensation jaune).

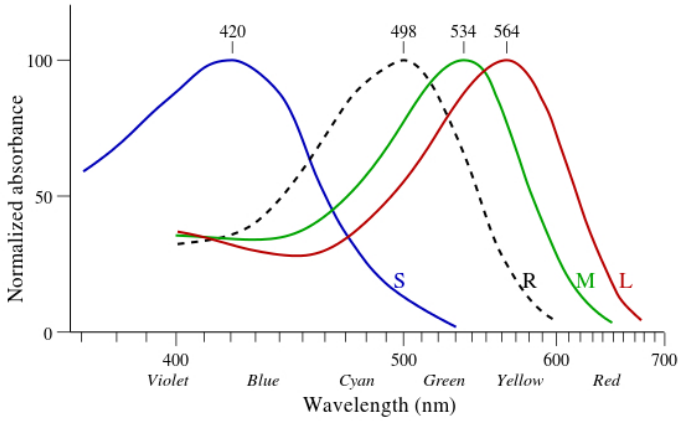


FIGURE I.5 – Courbes d'absorbance en fonction de la longueur d'onde. (S), (M) et (L) correspondent respectivement aux courbes pour les cônes bleus, verts et rouges et (R) correspond à la courbe relative aux bâtonnets [Bowmaker and Dartnall, 1980].

Dans la littérature scientifique anglo-saxonne, l'usage est de qualifier les cônes bleus de S (pour short), les cônes verts de M (pour medium) et les cônes rouges de L (pour long) en

référence à la longueur d'onde au maximum de sensibilité. Ces maximums de sensibilité sont par ailleurs différents de plusieurs nanomètres d'un individu à l'autre Ces cônes sont plus denses au centre de la rétine et deviennent de moins en moins denses à mesure qu'on se déplace vers la périphérie.

Les bâtonnets, eux, permettent la vision en niveau de gris dans le domaine mésopique et scotopique (vision nocturne). À l'inverse des cônes, leur densité est forte en périphérie et diminue au fur et à mesure qu'on se rapproche du centre de l'œil.

Cette disparité de fonctionnement entre cônes et bâtonnets explique notamment pourquoi lorsque la lumière est insuffisante, nous ne voyons pas ou peu les couleurs : seuls les cônes sont capables de percevoir les couleurs mais leurs performances sont beaucoup plus limitées que celles des bâtonnets pour des éclairages faibles.

L'appréciation des couleurs se fait en majeure partie dans la fovéa (très peuplée en cônes contrairement à la périphérie qui dispose principalement de bâtonnets "voyant" uniquement en nuance de gris). Donc, malgré les impressions suggérées par notre perception lorsque nous entrons dans une nouvelle pièce ou que nous regardons un nouvel objet, seule la vision centrale nous offre des informations sur la couleur. C'est grâce aux mouvements des yeux que nous pouvons avoir une appréciation globale de la couleur dans une scène. A contrario, la vision humaine est floue et peu colorée en périphérie. Dans cette région, le système visuel est surtout sensible aux mouvements grâce aux bâtonnets. Même si la zone de vision centrale où l'acuité est la meilleure est restreinte, il est possible de percevoir (de manière plus floue) dans l'espace périphérique.

L'activité oculaire, c'est à dire le mouvement par saccades-fixations des yeux, a donc un grand rôle [Lecas, 1992] puisque c'est elle qui va permettre d'ajouter des informations (couleurs, détails) sur des objets auparavant juste entraperçus en vision périphérique en permettant de déplacer la fovéa vers les zones intéressantes. Le processus itératif d'exploration visuelle est une succession de séquences saccades-fixations-saccades [Goldberg and Kotval, 1999].

Les saccades sont des mouvements rapides de l'œil qui servent à placer de manière inconsciente [Rodieck, 2003] une zone préalablement identifiée comme zone d'intérêt dans la zone fovéale. Les saccades sont des mouvements non interruptibles et très rapides (de l'ordre d'une dizaine de millisecondes par degré d'angle visuel). Un mécanisme correcteur [Fitts, 1992] permet d'ajuster la destination par de petites saccades qui suivent une saccade plus conséquente. Les saccades sont le fruit de la vision pré-attentive.

Les fixations s'intercalent entre les saccades et correspondent à la finalité de la vision pré-attentive qui devient attentive. C'est lors des fixations que les stimuli sont analysés dans la zone fovéale. Les fixations durent plus longtemps que les saccades : la durée totale d'une fixation est de l'ordre de quelques centaines de millisecondes. Durant ce laps de temps, trois phénomènes principaux se produisent [Goldberg and Kotval, 1999]. Premièrement, l'information visuelle est encodée pour pouvoir être traitée. Deuxièmement, le champ périphérique de la fixation courante est capturé afin de déterminer les nouvelles zones intéressantes. Enfin, la saccade suivante est préparée pour amener le centre du regard sur la fixation suivante. Une fixation n'est pas "fixe" au sens physique car durant la fixation, des micro-saccades se produisent à fréquence constante. Ce mécanisme sert à

réactiver les récepteurs visuels (les cônes et bâtonnets) qui envoient l'image au cerveau. Sans ce mécanisme, l'image s'estomperait progressivement.

Après avoir décrit rapidement le système physiologique du système visuel humain, il parait judicieux de se préoccuper des mécanismes attentionnels qui permettent de sélectionner les zones "intéressantes", zones qui déterminent quelle partie de l'image sera amenée à être traitée minutieusement grâce au déplacement de la zone fovéale de la rétine vers les zones concernées.

I.2.2 Mécanismes attentionnels chez l'être humain

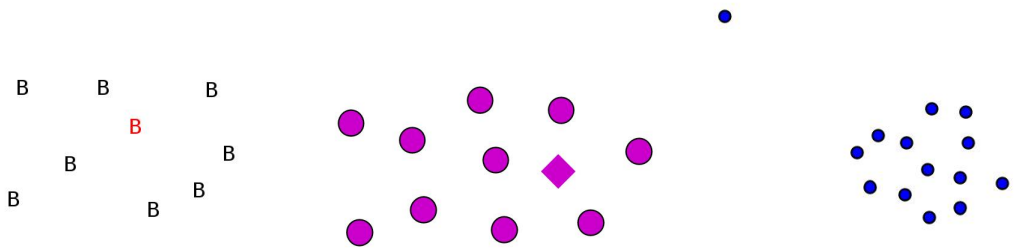
L'attention sélective chez l'homme joue un rôle important dans le processus de perception. C'est un mécanisme de "filtrage" des données perçues qui permet l'accès à des mécanismes plus complexes fournissant des informations détaillées [Delorme and Flückiger, 2003]. Un exemple très connu de ce phénomène, même s'il ne concerne pas la vision, est celui appelé *cocktail* : dans une salle se trouvent un ensemble de personnes discutant par petits groupes. Au sein d'un groupe, l'attention sélective permet de se concentrer sur la discussion du groupe pour pouvoir suivre la conversation, tout en se fermant aux autres conversations qui seront alors considérées comme du "bruit" à éliminer par le filtrage attentionnel. Il existe évidemment des processus similaires dans le domaine de la vision : nous sommes capables de filtrer les données pour nous concentrer uniquement sur des données particulières [Most et al., 2005, Simons and Chabris, 1999]. Les mécanismes attentionnels sont classés en deux grandes catégories [Fernandez, 2010] : les mécanismes exogènes et endogènes.

I.2.2.1 Mécanismes attentionnels exogènes

Nous reprenons ici la perception visuelle à titre pédagogique. En premier lieu, l'attention sélective peut être attirée par une localisation, un objet, un stimulus de manière indépendante du but recherché, des intentions de la personne. La cible "saute au yeux", on parlera également d'effet "pop-out". Il s'agit là de mécanismes attentionnels exogènes. Les processus ascendant ou "bottom-up" ou encore d'attention guidée par les caractéristiques désignent le même phénomène. Cette catégorie d'attention est rapide, involontaire et liée à l'arrivée d'un stimulus inattendu dans le champ visuel (distracteur).

La figure I.6 montre quelques exemples de ce type d'effets : sans demander à l'observateur de réaliser une quelconque tâche, son "attention" sera attirée par la lettre rouge au milieu des lettres noires, par le losange au milieu des cercles ou encore par le cercle en périphérie. Il est à noter que ces mécanismes ne sont pas propres à la perception visuelle : on retrouverait par exemple les mêmes effets pour la perception auditive. Un cri au milieu du silence attire l'attention indépendamment de ce qu'on souhaitait écouter. De même le son d'une sirène nous fera instinctivement réagir.

Des études [Posner, 1980, Wolfe and Horowitz, 2004] permettant d'évaluer l'efficacité relative de différents critères pour sélectionner un stimulus visuel montrent que la sélection reposant sur la localisation, la couleur, la taille, la brillance ou encore le mouvement donne de bonnes performances. La saillance d'un objet, c'est à dire l'attrait qu'il présente pour le mécanisme attentionnel de sélection, n'est pas intrinsèque à l'objet mais est plutôt dépendante du contexte [Nothdurft, 2000].



(a) attrait par la couleur (b) attrait par la forme (c) attrait par la localisation

FIGURE I.6 – Quelques exemples de situations où il est possible de prédire ce que verra en premier lieu l'observateur sans avoir besoin de lui avoir confié une tâche préalable

En effet, si nous regardons à nouveau les exemples précédents, la lettre rouge "saute aux yeux" parce que les autres lettres sont toutes d'une autre couleur. Si toutes les lettres de l'exemple étaient réécrites en rouge, aucune n'attirerait l'attention. Si déjà, les autres lettres étaient écrites en orange, la lettre rouge serait moins saillante. À l'identique dans les autres exemples, ce qui apparaît comme saillant c'est ce qui est différent dans un contexte donné, que ce soit par la forme, la couleur, la position ou autre.

Ces mécanismes exogènes sont couramment utilisés pour des raisons de sécurité par exemple. Ainsi les panneaux de signalisation routière ont des couleurs vives et sont conçus de manière à être particulièrement visibles : ils doivent "sauter aux yeux" des conducteurs. Semblablement, les sirènes de pompiers sont particulièrement repérables et ce même sous l'effet de la surprise. Ces phénomènes sont aussi utilisés par les publicitaires pour parvenir à mettre en avant leur produit : le client potentiel n'a pas nécessairement envie de regarder donc l'objectif est d'attirer son œil vers le produit de manière involontaire.

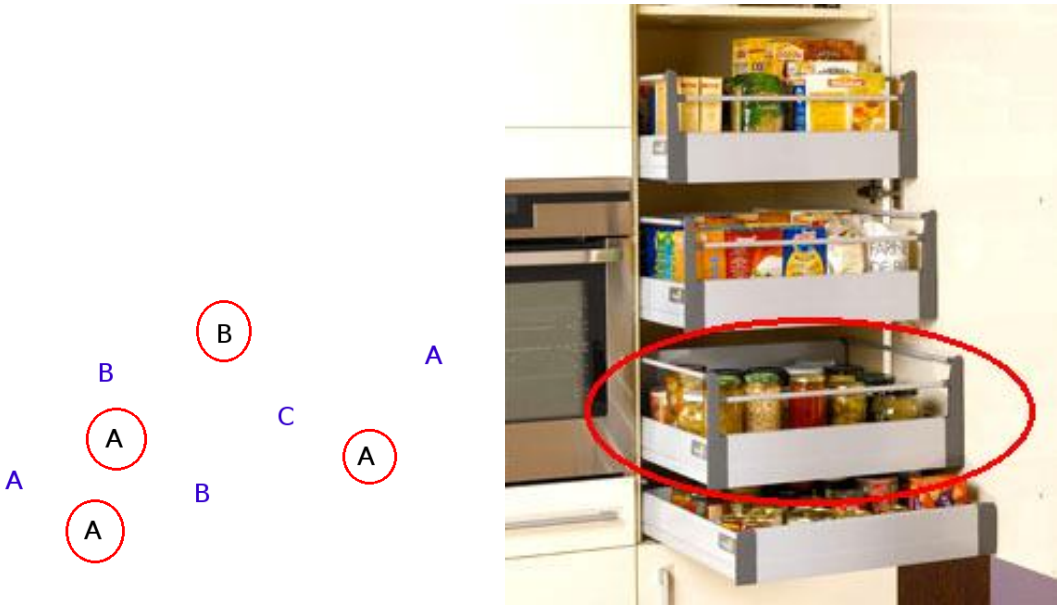
I.2.2.2 Mécanismes attentionnels endogènes

Si la sélection attentionnelle peut se faire de manière exogène (donc involontaire), il est aussi possible de l'orienter volontairement [Chun and Wolfe, 2001] vers un objet, une localisation ou un trait perceptif pertinent. Les mécanismes endogènes de l'attention correspondent à un acte délibéré, intentionnel du sujet visant à sélectionner un objet pour en améliorer le traitement. Dans le cas de la perception visuelle, la structure même de l'œil les rend nécessaires : pour voir le détail, il faut se concentrer (déplacer la fovéa) sur l'objet. Ces mécanismes sont aussi désignés sous le nom d'attention guidée par le but, de processus descendant ou "top-down". Le mécanisme attentionnel va s'orienter vers des cibles qui sont définies en fonction des besoins par rapport à une tâche en cours. Cette catégorie d'attention est plus lente, dure plus longtemps, mais permet d'être moins sensible aux distracteurs du fait de la focalisation de l'attention. La reconnaissance d'objets, par exemple, peut ainsi être plus efficace et pertinente [Riesenhuber and Poggio, 1999] : en focalisant l'attention dans une zone restreinte et/ou sur des caractéristiques spécifiques, il y a moins d'ambiguïtés entre l'objet recherché et le reste de la scène. Ce mécanisme attentionnel permet d'une manière générale d'accélérer les traitements de l'information perceptive [Delorme and Flückiger, 2003].

La figure I.7 montre quelques exemples de focalisation de l'attention visuelle en fonction des objectifs visés. Ainsi, dans la première sous image, si l'objectif est de trouver la lettre noire B, dans un premier temps, seules les lettres noires seront examinées. Dans la

seconde sous image, si on demande à l'observateur de chercher le bocal de flageolets dans le placard, il regardera plutôt dans le tiroir des bocaux. Un autre exemple connu, extrait de [Yarbus, 1967], est donné dans la figure I.8 où les saccades occulaires sont mises en avant en fonction de différents objectifs. Certaines zones sont donc en effet plus ciblées que les autres selon la tâche à accomplir.

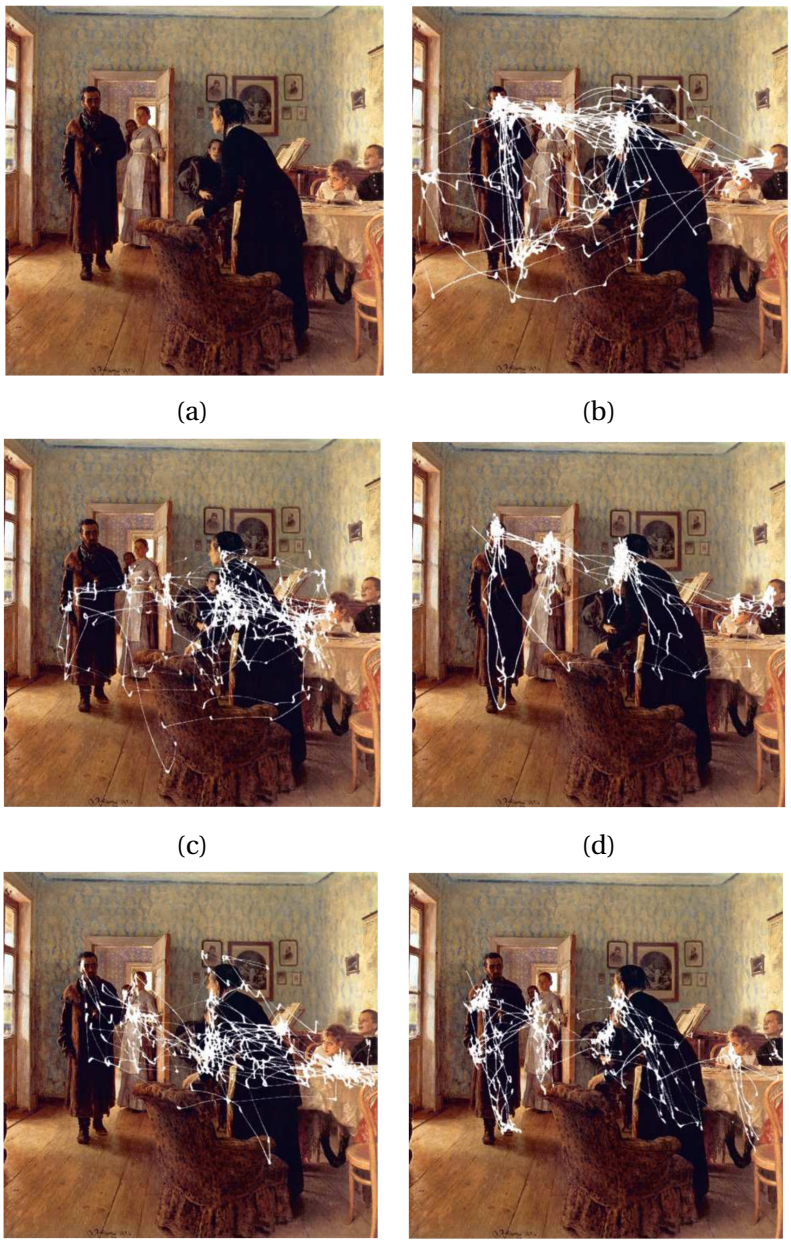
Il existe une différence fondamentale entre les mécanismes exogènes et les mécanismes endogènes. En effet, les premiers peuvent se faire de manière plus ou moins automatique, en tous cas passive alors que les seconds nécessitent d'avoir déjà des connaissances sur l'objectif de perception à atteindre : c'est à dire par exemple sur l'objet à rechercher dans le cas de la reconnaissance d'objets. Il faut une représentation interne de l'objet ou au moins des traits pertinents qui le caractérisent. Sans connaissance, il ne peut y avoir de mécanismes attentionnels endogènes. Ainsi, il n'est pas possible de réaliser une tâche efficacement sans représentation mentale de l'objectif. On peut par exemple penser aux très jeunes enfants à qui on demande de montrer dans un livre d'images "où est le crocodile ?" alors que le crocodile ne fait pas encore partie des animaux connus, il montrera alors un par un tous les objets non connus présents dans l'image (à supposer qu'il soit suffisamment patient) jusqu'à validation.



(a) (b)

FIGURE I.7 – Exemple de recherche orientée par le but : a) recherche de la lettre B noire, b) recherche du bocal de flageolets. En fonction de la consigne donnée par l'observateur, les zones de recherche apparaissant comme pertinentes sont encerclées en rouge. Seules ces zones seront traitées plus en détail jusqu'à validation de l'objectif.

Il faut également noter que l'objectif aura une importance primordiale pour valider la tâche : ainsi si l'objectif demandé est de trouver **la** lettre noire B, l'énoncé sous entend qu'il n'y en a qu'une, l'observateur s'arrêtera donc dès qu'il aura trouvé une lettre correspondant à la description dans l'ensemble des lettres. Si l'objectif demandé est de trouver **les** lettres noires B ou la lettre noire B **la plus** grande, l'observateur détaillera toutes les lettres noires afin de pouvoir répondre à la consigne.



(a) (b)

(c) (d)

(e) (f)

FIGURE I.8 – Impact de la tâche à effectuer sur le trajet oculaire lors de la visualisation d'une image par le même observateur : (a) image originale (tableau de Ilya Repin 1884 intitulé « An Unexpected Visitor») ; (b) aucune tâche n'est donnée ; (c) première tâche : estimer le niveau social des personnages ; (d) deuxième tâche : évaluer leur âge ; (e) troisième tâche : que faisaient les personnages avant l'arrivée du visiteur ? ; (f) quatrième tâche : souvenez-vous des habits portés par les différents personnages (Extrait de [Yarbus, 1967]).

Ceci met en exergue un autre élément essentiel dans un mécanisme d'attention pour la recherche en série : il s'agit du mécanisme d'inhibition de retour **IOR** (inhibition of return). L'IOR empêche l'observateur de revenir plusieurs fois sur le même élément de l'image : une fois examiné, l'élément ou la zone perd de son intérêt et ne sera plus retraitée à moins qu'une nouvelle information n'apparaisse. Bien que ce mécanisme soit présenté dans le paragraphe des mécanismes endogènes, il est également présent dans le cas des mécanismes exogènes.

I.2.2.3 Bilan des mécanismes attentionnels humains

En premier lieu, bien que nous traitions de manière générale de perception, pour des raisons de simplicité de représentation et de visualisation, nous avons surtout présenté (et présenterons encore) des exemples dans le domaine de la perception visuelle et orientés reconnaissance d'objets mais il faut garder à l'esprit que d'autres tâches sont possibles.

Nous avons vu qu'il existe deux mécanismes attentionnels différents qui permettent de guider l'attention vers des cibles données. Les deux vont utiliser des informations sur la forme, la couleur, la brillance ou la localisation d'un objet. Cependant, le premier type de mécanisme, dit exogène, est un mécanisme indépendant de la volonté de l'observateur qui sera principalement attiré par les éléments contrastant avec leur environnement. À l'inverse, le deuxième mécanisme dit endogène est un mécanisme volontaire orienté dans un but précis. Dans le cadre de la reconnaissance d'objets, ce mécanisme permettra d'utiliser les caractéristiques ou propriétés connues de l'objet pour le discriminer et ainsi l'identifier plus vite au milieu d'éventuels distracteurs.

Les deux mécanismes sont de manière évidente présents dans le cerveau humain [Buschman and Miller, 2007] mais il est difficile de quantifier dans quelle proportion ils interviennent que ce soit en terme de temps alloué à chacun d'eux, d'information apportée ou d'information utile fournie. S'il est évident que les mécanismes endogènes sont faits pour amener de l'information pertinente, les mécanismes exogènes sont aussi mis à contribution. Par exemple, un conducteur connaissant une route donnée ne s'attendra pas à voir un nouveau panneau (ajouté pour cause de travaux ou autre) mais appréciera tout de même d'apprendre que la route qu'il voulait emprunter est maintenant en sens interdit ou avec une autre limitation de vitesse ; les panneaux routiers étant volontairement construits pour être saillants et donc facilement visibles par un observateur.

Enfin, bien que les deux mécanismes aient été présentés séparément, il est admis qu'ils s'influencent mutuellement et ont des interactions. Par exemple, un panneau lumineux attirera de manière involontaire l'attention mais les indications affichées sur ce panneau (comme "attention au verglas") amèneront l'utilisateur à prêter plus attention aux caractères écrits et par la suite à la route (une tâche qui aurait potentiellement été ignorée sera attentivement scrutée pour déterminer si oui ou non elle présente un danger). L'attention involontaire peut donc moduler l'attention volontaire. L'inverse est également vrai : si quelqu'un cherche à trouver une personne particulière au milieu d'une foule, elle pourra avoir tendance à modifier involontairement la vision "réelle" des personnes qu'elle aperçoit pour la faire coïncider avec la personne recherchée, confondant les deux personnages jusqu'à ce que l'évidence du contraire devienne flagrante. L'avantage des mécanismes endogènes est qu'ils diminuent la prise en compte des distracteurs et permettent de se concentrer sur le but à accomplir mais l'inconvénient qui en découle et qu'ils peuvent en conséquence inhiber les mécanismes endogènes jusqu'à rendre "aveugle" à l'imprévu.

De plus, dans les deux cas, ces mécanismes aident à surmonter le problème de limitation de ressources et à savoir quels stimuli sont importants pour le système de perception [Dayan et al., 2000]. En effet, notre système visuel est intrinsèquement limité en capacité de traitement et notre environnement contient plus d'informations que nous ne pouvons en traiter simultanément. Pour pallier ce problème, notre système visuel s'est adapté en mettant en place des mécanismes, des stratégies pour réduire la quantité d'informations à traiter et ne sélectionner que les plus pertinentes (par rapport à un environnement donné avec des objectifs fixés).

Nous venons d'avoir un aperçu de la perception humaine et plus particulièrement de la perception visuelle. Nous allons maintenant nous pencher sur les algorithmes existants pour faire cette reconnaissance de manière artificielle et essayer de voir les liens qui existent avec la perception naturelle.

I.3 Perception artificielle

Bien que de nombreux progrès aient été réalisés au cours des dernières années, la perception artificielle reste un problème complexe et délicat. Là encore, nous illustrerons notre propos avec des exemples issus de la reconnaissance d'objets dans des images mais rien n'interdit de penser aussi à la reconnaissance d'objets sonores ou à de la perception pour réaliser une localisation de robot mobile par exemple.

Pour localiser et reconnaître des objets de nature variée dans des environnements potentiellement difficiles (environnement chargé avec de nombreux distracteurs, occultation partielle possible de l'objet recherché), les algorithmes de reconnaissance d'objets doivent être très flexibles et adaptables. De plus, l'environnement peut également être changeant (variation de l'illumination de la scène, objets en mouvement ou déplacement du ou des capteurs), complexifiant encore un peu plus le problème. D'ailleurs, les applications qui se sont le mieux développées sont souvent des applications industrielles qui présentent l'avantage d'être déployées dans des environnements contrôlés (éclairage constant, distance objet-capteur fixée, ...) ou dans des contextes spécifiques et pour lesquels un ensemble de caractéristiques influençant la détection est connu et fixé.

Les exigences de la reconnaissance d'objets : rapidité, invariance, robustesse au bruit qui sont aisées du point de vue perceptif humain sont justement celles qui posent le plus problème pour la reconnaissance artificielle. L'être humain peut gérer facilement des problèmes d'occultation partielle. Il peut même estimer la position d'un objet totalement occulté s'il l'a vu auparavant (soit en considérant que l'objet est fixe et que l'occultation est due à un distracteur qui s'est interposé entre lui et l'objet, soit en admettant une vitesse et un déplacement cohérents avec le passé impliquant que l'objet s'est déplacé derrière un autre). En outre, les problèmes de changement d'illumination (changement d'éclairage ou de conditions météorologiques en extérieur par exemple), de point de vue (un objet 3D selon l'angle auquel il est observé ne présentera pas les mêmes caractéristiques) ou encore de taille et de position (un être humain peut reconnaître aussi bien la statuette miniature d'un cheval que le cheval en taille réelle) complexifient grandement la tâche pour la reconnaissance artificielle. Pourtant ces mêmes problèmes sont courants et communs dans la vie quotidienne et sont aisément résolus par le système perceptif humain.

Pour résoudre ces problèmes, il existe deux grands types d'approches : les approches dites ascendantes ou bottom-up et les approches descendantes ou top-down. Les approches bottom-up ne se soucient pas dans un premier temps des objectifs à atteindre : elles vont d'abord récolter toutes les données capteurs disponibles pour ensuite les fusionner et les traiter afin d'en extraire de l'information qui pourra ou non être utile pour atteindre les objectifs. À l'inverse, les approches top-down partent des objectifs et vont sélectionner les informations utiles pour enfin descendre jusqu'au niveau capteur et chercher les données capteurs adaptées en fonction du but à atteindre. La figure I.9 permet de synthétiser ces deux modes de fonctionnement.

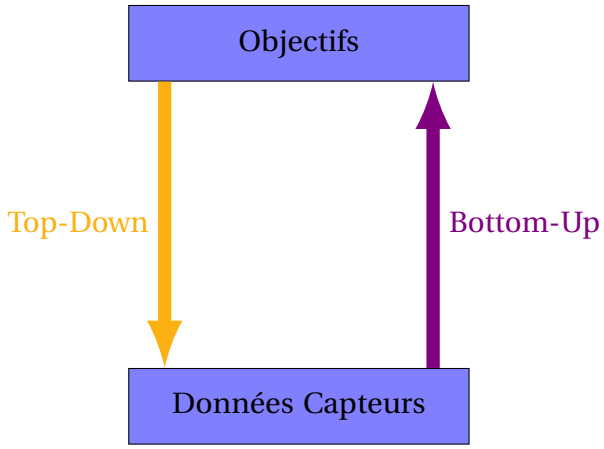


FIGURE I.9 – Schéma simple de fonctionnement des méthodes top-down et bottom-up

Dans les parties suivantes, sans prétendre à l'exhaustivité, nous allons étudier plus en détails ces deux modes de fonctionnement.

I.3.1 Méthodes bottom-up

Le principe des méthodes bottom-up est souvent le suivant :

— extraction de primitives de bas ou de moyens niveaux tels que des coins ou des segments

— rassemblement des primitives selon des règles de construction fixées

— génération d'hypothèses

— évaluation à l'aide d'une fonction de coûts

I.3.1.1 Imitation des mécanismes attentionnels exogènes

Les méthodes bottom-up font échos aux mécanismes attentionnels exogènes. En effet dans les deux cas, l'objectif est de partir des données et de faire jaillir une information (qui "saute aux yeux") à partir de ces données. Il existe en conséquence de nombreuses applications dont l'objectif n'est pas de reconnaître un objet précis mais d'identifier les objets les plus saillants, c'est à dire les objets qui naturellement attirent le regard d'un observateur. Le but est alors d'imiter au mieux les mécanismes attentionnels exogènes en parvenant à mettre en exergue de manière automatique par un ordinateur, les éléments saillants.

La plupart des modèles informatiques qui imitent les mécanismes exogènes de l'attention ont des structures similaires qui sont adaptées de la *théorie d'intégration des caractéristiques* de Treisman and Gelade [Treisman and Gelade, 1980] et du modèle de recherche orientée de Wolf et al [Wolfe et al., 1989]. Koch et Ullman [Koch and Ullman, 1985], les premiers, proposent une architecture biologiquement plausible pour l'attention visuelle. La principale idée est de calculer différentes caractéristiques bottom-up telles que la couleur, l'orientation ou la brillance en parallèle et de fusionner les cartes de caractéristiques ainsi obtenues en une carte 2D usuellement appelée **carte de saillance**.

De manière un peu plus détaillée, ces algorithmes suivent généralement les étapes suivantes : en premier lieu, une ou plusieurs pyramides d'images sont calculées à partir de l'image originale afin de permettre par la suite l'extraction de caractéristiques à différentes échelles. Puis les différentes cartes de caractéristiques sont calculées. Les caractéristiques les plus communément utilisées sont l'intensité, la couleur et l'orientation. Chacune de ces caractéristiques peut être découpée en plusieurs sous caractéristiques comme rouge, bleu et vert pour la couleur. Puis des mécanismes de gradients orientés ou de différences de gaussiennes sont utilisés pour calculer les cartes de caractéristiques grâce aux contrastes présents dans l'image : la valeur moyenne d'une région centrale et comparée avec la valeur moyenne des régions alentours et donc seules les régions singulières sont mis en exergue par ces procédés. Les cartes de caractéristiques sont ensuite sommées pour former des cartes d'évidence (**conspicuity map**) qui sont ensuite normalisées et combinées ensemble avec différents poids pour chaque carte intermédiaire afin d'obtenir la carte de saillance finale. Cette carte de saillance se présente le plus souvent sous la forme d'une image en niveau de gris où la brillance d'un pixel indique son niveau de saillance. La figure I.10 extraite de [Itti and Koch, 2001] illustre ce mode de fonctionnement.

Cette carte de saillance peut être considérée comme la sortie finale de ces algorithmes puisqu'elle détermine la saillance pour chaque région de l'image. La figure I.11 montre quelques exemples de cartes de saillance obtenues à l'aide de différents algorithmes. Ainsi, la carte de saillance résultant de [Itti et al., 1998] renvoie plutôt des zones assimilables à des points de fixation du regard humain tandis qu'à l'autre extrémité, les auteurs de [Achanta et al., 2009] s'intéressent plutôt à une carte de saillance mettant en exergue un objet en entier à des fins de segmentation.

Certaines applications [Itti et al., 1998, Le Meur et al., 2006] préfèrent travailler avec les régions les plus saillantes en les ordonnant pour reconstituer une trajectoire du regard (afin d'imiter les saccades-fixations du système visuel humain). Les régions sélectionnées seront alors celles correspondant à un maximum local. Une approche *Winner-Take-All* ainsi qu'un mécanisme d'inhibition de retour assurent que tous les maximums sont traités et évitent qu'il n'y ait une stagnation sur la région la plus saillante.

Un des aspects importants de ces systèmes est la manière dont les différentes cartes intermédiaires sont fusionnées. En effet, il n'y a pas de modèle certifié conforme au fonctionnement du cerveau humain et en conséquence différentes méthodes peuvent être utilisées. Pour la fusion, des poids sont attribués aux différentes cartes, les poids donnant alors une indication de l'importance relative d'une caractéristique par rapport à une autre. Avant l'attribution des poids, une étape de normalisation permet de rendre comparables des cartes qui a priori ne le sont pas car obtenues avec des mécanismes différents et ne représentant pas les mêmes modalités.

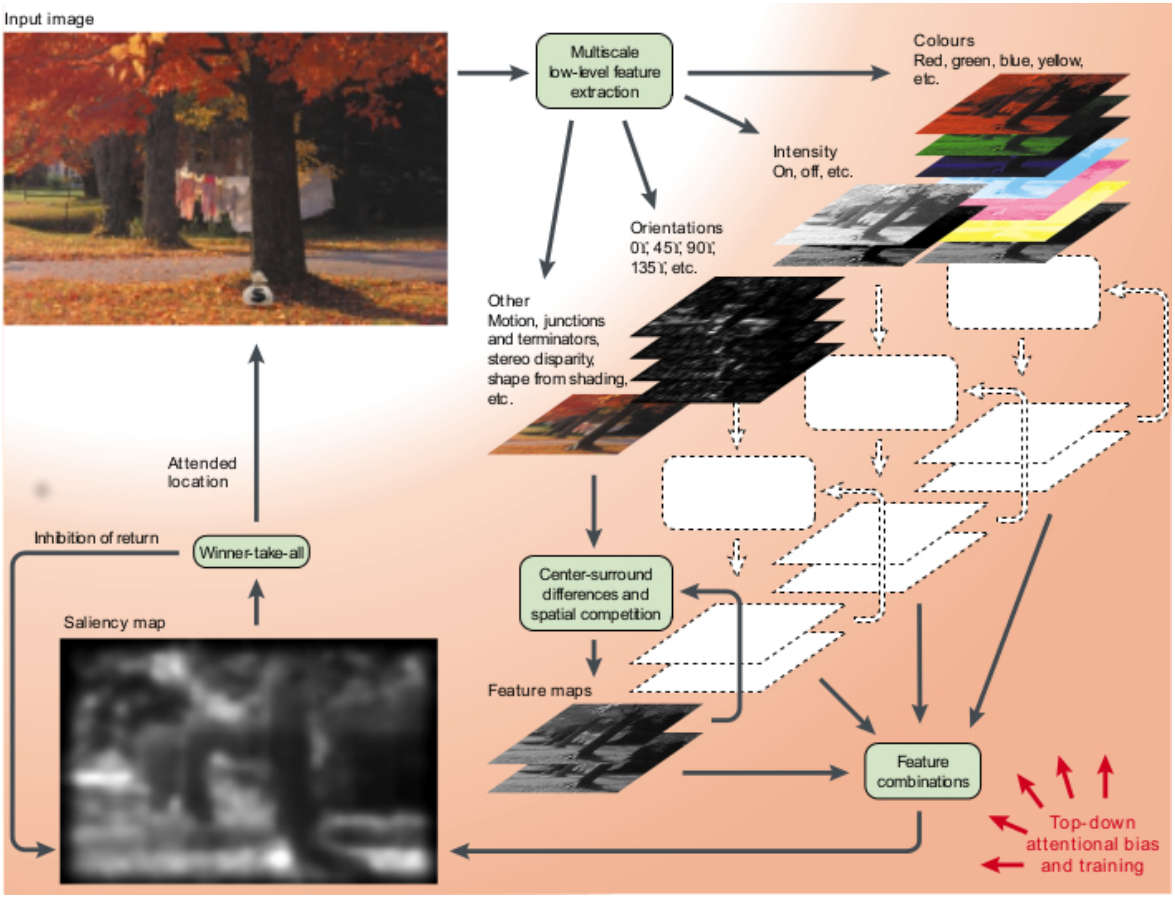
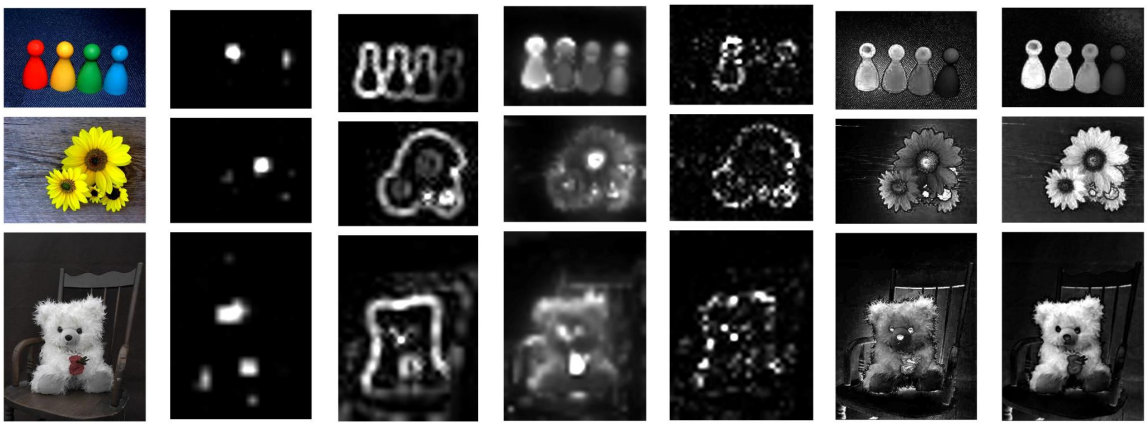


FIGURE I.10 – Structure basique pour la création de carte de saillance extraite de [Itti and Koch, 2001]



(a) Original (b) IT (c) MZ (d) GB (e) SR (f) AC (g) IG

FIGURE I.11 – Quelques exemples de cartes de saillance. La première colonne représente les images originales, les colonnes suivantes sont les cartes de saillance obtenues à l'aide des algorithmes de b) [Itti et al., 1998], c) [Ma and Zhang, 2003], d) [Harel et al., 2007], e) [Hou and Zhang, 2007], f)[Achanta et al., 2008], g)[Achanta et al., 2009]. Les images sont issues de [Achanta et al., 2009].

Clark and Ferrier [Clark and Ferrier, 1988] sont parmi les premiers à avoir implémenté un système d'attention basé sur le modèle de Koch and Ullman [Koch and Ullman, 1985]. La première étape de leur système d'attention consiste à calculer en parallèle des cartes pour différentes caractéristiques, chaque pixel de ces cartes indique alors la présence ou l'absence de la caractéristique recherchée. Les caractéristiques les plus simples sont liées à la couleur ou à l'orientation de lignes tandis que d'autres représentent des données plus complexes comme de la texture ou des phénomènes d'inhibition de la part de régions voisines afin de permettre de différencier les régions à fort contraste. Ensuite, pour obtenir la carte de saillance finale, les cartes précédentes sont multipliées par un gain et sommées entre elles. Déjà apparaissent les premières grandes idées des cartes de saillance comme le calcul en parallèle de différentes caractéristiques, leur modulation par des poids et leur sommation finale.

Milanese et ses collègues [Milanese et al., 1994] introduisent d'autres concepts toujours utilisés aujourd'hui comme ceux impliquant de faire la différence entre les valeurs moyennes des pixels d'une région centrale et ceux des régions alentours à l'aide de différence de gaussiennes orientés (**difference of oriented gaussian (DOG)**). Comme caractéristiques, ils ont choisi d'utiliser entre autres les contrastes rouge-vert, les contrastes bleu-jaune ainsi qu'une intensité achromatique obtenue en sommant les trois canaux de couleurs RGB. Le mouvement apparaît aussi comme une caractéristique à prendre en compte.

D'autres dérivés du modèle de Koch et Ullman sont les travaux de Itti et al [Itti et al., 1998, Itti and Koch, 2000]. Le modèle d'Itti fait partie des modèles les plus connus des cartes de saillance. Il est souvent utilisé comme base de comparaison. Quatre types de cartes de caractéristiques sont calculés sur l'image : les contrastes rouge-vert, les contrastes bleu-jaune, les contrastes en luminance et les variations d'angles de gradient. Une des plus grandes nouveautés de cet algorithme est de proposer le calcul de ces cartes de caractéristiques à différentes échelles. Toutes les cartes ainsi calculées sont ensuite normalisées avant d'être sommées (avec différents poids). Les auteurs ont proposé des améliorations comme une normalisation itérative [Itti and Koch, 2001] ou encore l'ajout de composantes top-down [Navalpakkam and Itti, 2006].

Beaucoup d'autres auteurs ont suggéré des améliorations au modèle de base proposé dans les travaux d'Itti. Certains ont cherché à le rendre plus robuste : Draper and Lionelle [Draper and Lionelle, 2003] proposent l'algorithme SAFE (**selective attention at front end**) qui modifie l'algorithme initial afin de le rendre plus stable aux transformations géométriques telles que les translations, les rotations et les réflexions. Dans l'optique de diminuer les réglages à faire selon les objets, dans [Vikram et al., 2011a, Vikram et al., 2011b], un parcours aléatoire de l'image est effectué, rendant la carte de saillance plus robuste en permettant de prendre en compte des régions voisines de plus ou moins grande taille.

Pour optimiser les traitements, Harel et al [Harel et al., 2007] utilisent des algorithmes de graphes afin de calculer efficacement leur carte de saillance fondée sur l'utilisation d'une mesure de dissimilarité. Dans [Xie and Lu, 2011], une approche par raffinement utilisant un modèle bayésien est mise en œuvre afin d'éviter la recherche multi-échelles tout en permettant de trouver des objets de tailles différentes. À l'inverse, dans [Sahli et al., 2012], les auteurs profitent des caractéristiques de taille de l'objet pour en déduire des

propriétés intéressantes à rechercher.

Enfin, le choix des différentes caractéristiques a été un axe de recherche pour de nombreux auteurs. Le Meur et al [Le Meur et al., 2005] conserve l'utilisation de caractéristiques liées à la couleur mais choisissent d'en changer la représentation de l'espace de couleurs : ils utilisent l'espace des couleurs de Krauskopf qui est une représentation en luminance-chrominance des couleurs proche de la vision humaine. De nouvelles caractéristiques peuvent aussi être prise en compte comme la symétrie [Achanta and Süsstrunk, 2010, Privitera and Stark, 2000], les courbes [Valenti et al., 2009], le contraste global dans l'image [Cheng et al., 2015]. De manière un peu plus originale, les auteurs de [Hou and Zhang, 2007] s'intéressent aux caractéristiques non pas des objets saillants mais de ce qui constitue l'arrière plan de l'image et s'en servent pour reconstituer une carte de saillance. Les auteurs de [Urban et al., 2010] étudient le type de caractéristiques en fonction des environnements (environnements naturels ou au contraire environnements fabriqués). Ils montrent que des caractéristiques de bas à moyens niveaux conviennent dans l'ensemble mieux aux scènes naturelles tandis que des caractéristiques de moyen à haut niveaux donnent des meilleurs résultats pour les autres.

Il existe ainsi une multitude de modèles pour créer des cartes de saillance. Les objectifs recherchés peuvent être différents. Ainsi, Achanta et al [Achanta et al., 2008, Achanta et al., 2009] ont conçu leur carte de saillance dans le but de faciliter la tâche de segmentation des objets, c'est également le cas dans les travaux de [Valenti et al., 2009]. Les auteurs de [Liu et al., 2007, Sahli et al., 2012] orientent leur modèle pour la reconnaissance d'objets saillants tandis que ceux de [Hong et al., 2015, Zhang et al., 2015] proposent d'utiliser les cartes de saillance pour le suivi d'objets. La compression d'images peut être améliorée à l'aide de carte de saillance en permettant de diminuer davantage les zones inintéressantes et de conserver une meilleure qualité pour les zones regardées plus attentivement [Itti, 2004, Chenlei Guo and Liming Zhang, 2010]. Dans le domaine de la robotique, les auteurs de [Siagian and Itti, 2009] utilisent les objets saillants dans l'environnement pour aider à la navigation d'un robot. Dans [Clark and Ferrier, 1988], un robot est contrôlé afin d'orienter ses caméras pour suivre l'objet saillant. La saillance peut aussi servir à sélectionner des sous-images pertinentes [Le Meur et al., 2006, Marchesotti et al., 2009]. Enfin, trouver des régions d'intérêt peut aider à déterminer le niveau de détails approprié pour styliser ou simplifier des modèles afin de les rendre plus compréhensibles comme dans [DeCarlo and Santella, 2002, Grabli et al., 2004].

De manière générale, nous avons vu différentes méthodes permettant de calculer des cartes de saillance censées aider à modéliser les mécanismes exogènes humains. La plupart de ces méthodes sont fondées sur le modèle de Koch et Ullman [Koch and Ullman, 1985] et diffèrent par les caractéristiques choisies ainsi que les manières de les fusionner. Selon l'application recherchée et l'environnement, certaines seront plus pertinentes que d'autres.

Cependant, il existe également des méthodes de type bottom-up permettant de trouver dans une image des objets prédéterminés : plutôt que de faire remonter les informations saillantes de l'image, on s'attache davantage à réaliser l'appariement d'un modèle avec l'information perçue. En effet, les cartes de saillance sont particulièrement bien adaptées pour faire surgir de l'information non prévue et inattendue mais souvent elles ne permettent pas de savoir à l'avance quel objet sera perçu comme saillant et donc sera mis en valeur par la carte de saillance. Il peut alors être utile d'avoir un autre processus permettant de reconnaître des objets donnés, c'est ce que nous verrons dans les paragraphes suivants.

I.3.1.2 Modèles basés sur l'apparence

Nous allons aborder ici les techniques par **appariement de modèles** aussi appelées techniques **basées sur l'apparence**. Dans ces techniques, l'objet est typiquement représenté par son apparence c'est à dire par les données perçues qu'il est possible d'obtenir de lui plutôt qu'à partir d'une modélisation abstraite de cet objet.

I.3.1.2.a Principe des modèles basés sur l'apparence

Un modèle de l'objet est supposé connu. La plupart du temps, ce modèle est construit à l'aide d'une base d'apprentissage constituée d'exemples positifs (l'objet est présent) et d'exemples négatifs (il s'agit d'autres objets, de fragment de l'objet recherché ou encore d'éléments du décor). La plupart du temps, il est nécessaire de disposer d'une grande base d'apprentissage. À l'aide de ce modèle appris, il est ensuite possible d'appeler une fonction de classification qui renvoie un résultat indiquant si l'objet cherché est présent dans le nouvel échantillon présenté (de même type que ceux présents dans la base d'apprentissage).

I.3.1.2.b Modèle

Nous différencierons deux grandes catégories de modèles : les modèles implicites et les modèles explicites. Ces derniers sont des modèles énoncés formellement, connus à l'avance et descriptibles. Par exemple, une ligne peut être représentée par l'équation suivante : $ax + by + c = 0$. Les inconnues de la ligne sont a, b, c et tout point (x, y) appartenant à la ligne vérifie l'équation précédente. Il n'y a donc aucune ambiguïté sur la description, elle est énoncée préalablement et ne prête pas à discussion. Il est possible de verbaliser le savoir contenu dans le modèle. Ce type de modèle est par exemple utilisé dans l'algorithme du RANSAC [Fischler and Bolles, 1981] et ses variantes [Chum and Matas, 2005, Sattler et al., 2009, Rabin et al., 2010] où un modèle explicite doit être donné en entrée pour permettre de comparer les données sélectionnées au modèle instancié construit. Ainsi pour l'exemple de la droite, il faudra calculer l'erreur entre la droite instanciée (des valeurs ont été attribuées aux paramètres a, b, c) et les points censés la constituer. Ce type de modèle n'est pas nécessairement facile à construire puisqu'il exige non seulement des connaissances précises sur l'objet à modéliser mais également une forte capacité d'abstraction et potentiellement une représentation complexe au risque sinon d'être incomplète. En contrepartie de cette création difficile, ces modèles ont un ensemble de paramètres parfaitement connu dès le départ. Ils sont adaptés pour représenter des modèles déformables et particulièrement des modèles en parties (voir le § I.3.2.1) puisqu'ils permettent d'expliquer et de prévoir dans quelle mesure différentes parties de l'objet peuvent avoir changé de configurations. Un modèle explicite facilite également la gestion d'objets ayant des apparences très variables puisque les variations d'apparence seront liées à des positions particulières de l'objet. Enfin, un modèle explicite est plus souple pour gérer les problèmes d'occultation : il est plus facile en connaissant le modèle de prévoir et de gérer qu'une ou plusieurs parties de l'objet puissent ne pas être visibles.

Les modèles implicites, eux, ne sont pas énoncés formellement mais découlent "naturellement" de quelque chose. Ils peuvent être définis comme correspondant "*aux connaissances que nous développons à notre insu sans être capables de les verbaliser*" [Simoës-Perlant and Largy, 2008]. La connaissance comprise dans ces modèles est donc difficile à caractériser ou à exprimer verbalement mais permet néanmoins l'acquisition de connaissances complexes. Un exemple type de ces modèles implicites est l'acquisition du langage

maternel chez un enfant. Ce dernier apprend à maîtriser un outil complexe sans aborder cette tâche sur un mode analytique et sans pouvoir expliquer ce qu'il apprend exactement. Ainsi un jeune enfant sera probablement incapable d'expliquer pourquoi il dit "un cheval" et des "chevaux" ou encore "je suis" et "il est". Les cartes de saillance précédentes appartiennent plutôt à cette seconde catégorie de modèles. Un objet peut ressortir saillant en raison de sa taille, de sa forme, de sa couleur ou autre. Néanmoins le résultat renvoyé pour l'objet consistera uniquement en un indice de saillance sans que ses caractéristiques soient clairement identifiées.

Les modèles implicites sont typiquement adaptés pour être construits à partir de nombreux exemples, déduisant de ceux-ci les caractéristiques de ce qui est cherché sans nécessairement pouvoir les extraire facilement du modèle. Ils permettent aisément de représenter des objets d'une même classe présentant de légères différences sous un même modèle : tous les objets sont donnés comme des exemples de la classe (et donc du modèle) à construire et à identifier. Du fait de la représentation implicite, il n'est pas nécessaire d'être un expert pour connaître le modèle, ni d'être capable de donner toutes les composantes utiles d'un objet. Elles seront automatiquement déduites des exemples proposés. La construction de ces modèles est donc conceptuellement beaucoup plus simple même si elle nécessitera une base d'apprentissage. Par ailleurs, elle est très facile à mettre en œuvre. En contrepartie, il est difficile d'extraire des informations à partir du modèle ainsi créé et les modèles peuvent avoir des difficultés à représenter une classe comportant des objets trop différents. Dans la suite de cette partie, nous nous intéresserons donc à ces modèles implicites.

I.3.1.2.c Apprentissage

Pour construire un modèle implicite, il nous faut donc une base d'apprentissage comprenant de nombreux exemples. En règle générale, les paramètres calculés pour l'apparence sont obtenus en prenant les données (comme par exemple une image) comme un tout. En conséquence, la plupart des bases d'apprentissage nécessitent d'avoir des exemples avec l'objet présentant de petites variations (liées au changement d'illumination ou à un léger changement de point de vue par exemple) et sans occultation.

La forme générale de la base d'apprentissage est le plus souvent (x_i, y_i) où x_i est le vecteur de caractéristiques et y_i la réponse attendue (*objet* ou *pas objet* ou tout couple de valeurs différentes auquel aurait été attribué une signification identique : par exemple -1 correspond à *pas objet* et 1 à *objet*) [Schölkopf and Smola, 2002a]. x_i contient les mesures des propriétés de l'objet : il peut contenir des informations sur la couleur, la géométrie de l'objet, la réponse à des descripteurs spatiaux tels que des ondelettes de Haar. x_i doit être le plus représentatif possible des éléments importants dans l'apparence de l'objet.

La construction de cette base d'apprentissage n'est pas une tâche aisée. Elle demande le plus souvent une annotation manuelle de tous les exemples, ce qui peut être une tâche très fastidieuse s'il y a beaucoup d'exemples. Il faut suffisamment d'exemples positifs pour pouvoir représenter la variabilité de l'objet et suffisamment d'exemples négatifs pour éviter de tout classer dans la catégorie *objet*. Les temps pour apprendre le classifieur sont souvent longs également. On parle du fléau de la dimensionnalité [Jain et al., 2000] : il faut de l'ordre de dix fois plus d'exemples que de dimensions de classification (liées à la taille minimum du vecteur x_i pour bien modéliser l'objet) .

Dans le cadre de la reconnaissance artificielle par vision, il faut cependant noter qu'il est de plus en plus aisé de trouver des bases déjà annotées [Caltech Database, , The PASCAL Visual Object Classes, , LFW Face Database,] pour un nombre croissant de classes telles que les visages, les voitures etc. Cela présente l'avantage de permettre de comparer les résultats de différents algorithmes sur les mêmes images afin d'avoir une vision la plus objective possible sur leurs qualités et leurs défauts en fonction du contexte. Cela permet également un gain de temps considérable en mutualisant les fastidieuses annotations manuelles : elles sont faites une fois et profitent ensuite à un grand nombre de chercheurs.

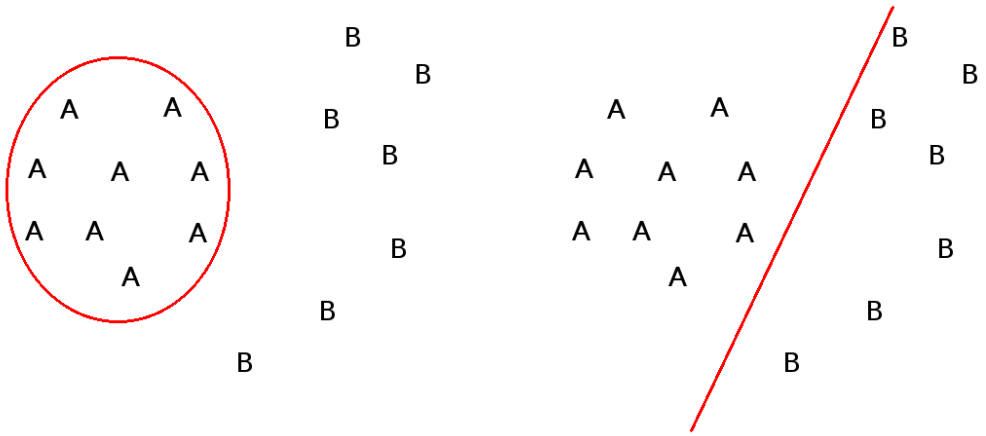
Pour l'apprentissage, une étape de classification comme présentée dans le § I.3.1.2.d est souvent utilisée. Un premier modèle (grossier) est construit puis affiné à l'aide de nouveaux échantillons : le modèle est corrigé en fonction des erreurs de classifications faites sur l'ensemble des échantillons. Cette étape peut être répétée plusieurs fois lors de la phase d'apprentissage.

I.3.1.2.d Classification

Une fois le modèle appris, l'objectif est de retrouver dans un échantillon à tester (une imagette par exemple) si l'objet ainsi modélisé est présent et à quelle position. Pour cela, le plus souvent, une fenêtre de taille donnée est déplacée dans tout l'échantillon. Pour chaque position de la fenêtre, un test est réalisé pour savoir si l'objet est présent. Le test d'une fenêtre peut être vu comme une fonction de classification entre deux classes : *objet* et *non objet* ; ou encore comme une fonction de corrélation entre une image de référence (de la base d'apprentissage) et une sous-image de l'échantillon de test.

Les méthodes de classification peuvent être divisées en deux sous catégories : les méthodes génératives et les méthodes discriminatives. Les premières incluent, par exemple, l'Analyse Discriminante Linéaire [McLachlan, 2004] et les classifieurs de Bayes Naïfs [Zhang, 2004]. Les secondes incluent la méthode des k plus proches voisins, les arbres de décisions [Quinlan, 1986], les machines à vecteur support plus connues sous le nom de **SVM** (Support Vector Machine) [Cristianini and Shawe-Taylor, 2000]. La figure I.12 montre un exemple simple illustrant les différences entre une classification générative et discriminative sur un même jeu de données. Dans les modèles génératifs, seule la distribution des données positives est modélisée. Un nouvel échantillon sera ensuite comparé au modèle et la classification se fera selon le degré de ressemblance entre l'échantillon et les exemples positifs du modèle. Dans les méthodes discriminatives, la distribution des exemples positifs pour une classe est comparée avec les exemples négatifs (ou également avec les exemples positifs des autres classes). Les fonctions de décisions sont ensuite construites afin de séparer de manière la plus optimale possible les différentes classes. Les modèles discriminants permettent d'identifier les caractéristiques (ou les combinaisons de caractéristiques) pertinentes pour différencier les classes et obtiennent le plus souvent de meilleurs résultats [Csurka et al., 2004, Goutte et al., 2004]. Cependant, cela dépend grandement de la base d'apprentissage [Ng and Jordan, 2002] : en effet le manque d'exemples négatifs adéquats peut distordre les frontières et amener à de mauvaises classifications.

Les **SVM** [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2002b, Vapnik, 1999] ou ses extensions telles que les SVM structurelles [Blaschko and Lampert, 2008, Tsochantaridis et al., 2004] sont des méthodes très connues. Les plus simples sont les SVM linéaires. Le score est alors calculé simplement en faisant le produit scalaire entre le vecteur caractéristique de la classe et le vecteur créé à partir des données d'entrée. Dalal et Trigs



(a) exemple de séparation par méthode générative

(b) exemple de séparation par méthode discriminative

FIGURE I.12 – Illustration des différences entre méthodes génératives et discriminatives sur un exemple simple

[Dalal and Triggs, 2005] se servent par exemple de ces méthodes pour la détection de personnes en utilisant des histogrammes de gradients orientés (**HOG**) pour la représentation de la classe. Néanmoins, ces modèles peuvent avoir des difficultés pour bien catégoriser des classes dont les objets présentent de fortes variabilités. Il existe cependant des méthodes à noyau non linéaires qui peuvent donner de bons résultats pour la catégorisation [Van Nguyen et al., 2013].

D'autres modèles possibles sont les modèles Gaussiens (ou les mixtures de Gaussiennes). Ils font partie des méthodes génératives. Ils stockent un ensemble de vecteurs propres et les valeurs propres liées, ce qui permet de représenter la variance de la classe. Par exemple, en 2D, tous les échantillons (dans l'exemple des points) qui sont dans l'ellipse de la figure I.12a sont considérés comme appartenant à la classe A. Classer un nouvel échantillon consiste alors simplement à savoir s'il se trouve ou non dans l'ellipse. Cette dernière peut être caractérisée par des vecteurs propres et des valeurs propres. Pour des dimensions supérieures, le principe reste le même. Pour avoir une méthode non plus générative mais discriminative, il est possible de créer un autre ensemble de gaussiennes représentant les exemples négatifs et de déterminer s'il est plus probable que l'échantillon appartienne à l'un ou l'autre des ensembles (est-ce qu'il est plus proche du modèle "objet" que du modèle "non-objet"). Ces modèles ont été par exemples utilisés dans la reconnaissance de visages [Kirby and Sirovich, 1990, Turk and Pentland, 1991].

Viola et Jones [Viola and Jones, 2001, Viola and Jones, 2004] utilisent une cascade d'arbres de décision entraînés en utilisant l'algorithme d'Adaboost. Des arbres superficiels sont utilisés comme classifieurs faibles, ils évaluent la réponse à des ondelettes de Haar pour chaque noeud et essaient de séparer les exemples positifs des exemples négatifs. Chaque arbre renvoie une décision binaire ("oui" ou "non"). Les différents arbres sont entraînés itérativement, l'un après l'autre. À chaque ajout d'un arbre à l'ensemble des arbres déjà entraînés, les exemples de la base d'apprentissage sont repondérés en prenant en compte les erreurs de classification de l'ensemble courant : l'objectif est de concentrer les efforts de classification du nouvel arbre sur les exemples qui ne sont pas encore bien classifiés. La sortie de l'ensemble est la somme des sorties de chaque arbre pondérées

par leur taux d'erreurs individuel. Afin d'augmenter la rapidité d'exécution, des groupes d'arbres sont formés pour constituer une étape de la cascade. Chaque étape de la cascade est entraînée de manière à assurer un certain taux de rappel et une précision minimum. Le but est d'éliminer le plus rapidement possible les parties non intéressantes afin d'appeler la suite de la cascade uniquement sur les régions intéressantes (voir figure I.13). En effet, à chaque étape de la cascade si pour un échantillon donné la classification est négative, la cascade est arrêtée et les tests plus approfondis ne sont pas lancés.

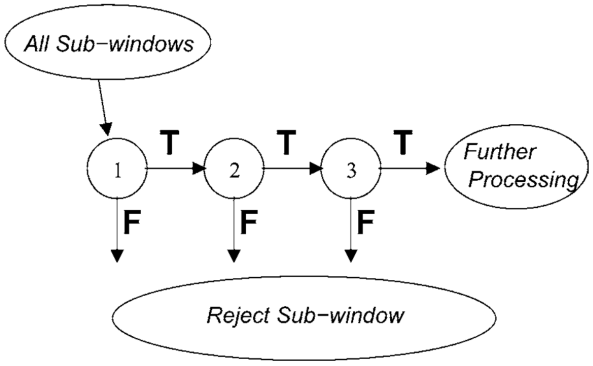


FIGURE I.13 – Schéma simple d'une cascade de détection extrait de [Viola and Jones, 2004]. Une série de classifieurs est appelée sur chaque fenêtre de classification. Les **F** (False) indiquent un résultat négatif : les classifieurs appelés considèrent que la fenêtre traitée ne représente pas l'objet cherché. La fenêtre considérée est alors rejetée. À l'inverse, les **T** indiquent un résultat positif : la fenêtre sera alors examinée par d'autres classifieurs pour confirmer qu'il s'agit bien de l'objet cherché. Les classifieurs initiaux éliminent un grand nombre d'exemples négatifs avec très peu de traitements. Les étapes suivantes éliminent d'autres exemples négatifs mais au prix d'un plus grand temps de calcul. Après quelques étapes, le nombre de fenêtres non encore classées a diminué drastiquement.

Les extensions incluent les *soft cascades* [Bourdev and Brandt, 2005] qui permettent de stopper la cascade après chaque arbre individuel (au lieu de traiter tout le groupe) ou les *crosstalk cascades* [Dollár et al., 2012] qui permettent de ne pas appeler les détecteurs dans les zones de l'image qui ne sont pas prometteuses.

D'autres méthodes utilisées pour la classification sont les réseaux de neurones [Krizhevsky et al., 2012]. Ces derniers ont été créés en s'inspirant du cerveau humain connu pour ses formidables capacités d'analyse et adaptabilité qui lui permettent entre autres de prendre en compte de nouveaux événements et d'apprendre de ses erreurs. Le neurone, ou cellule nerveuse, est une cellule excitable constituant l'unité fonctionnelle de base du système nerveux. Des chercheurs se sont donc penchés sur son fonctionnement afin d'essayer de le reproduire au moins en partie. C'est de ces recherches que sont nés les réseaux de neurones. Un réseau de neurones est constitué de plusieurs entités reliées entre elles par des actions soit de stimulation soit d'inhibition. Les réseaux de neurones ont été utilisés avec succès pour la reconnaissance d'écriture [Cho, 1997], pour de l'aide médicale [Dahab et al., 2012, Sankari and Adeli, 2011], dans des problèmes de classification [Krizhevsky et al., 2012, Oquab et al., 2014].

Quelle que soit la méthode utilisée, l'avantage de la classification est d'avoir une utilisation particulièrement simple puisqu'il suffit de parcourir l'image et d'appeler la fonction de classification dans la fenêtre glissante. Les traitements des données sont réguliers (pour

une même image, les temps de traitements seront toujours du même ordre de grandeur) et ce sont des traitements assez simples à faire pour un ordinateur ce qui se prête donc bien à leur utilisation à l'heure actuelle. De plus, comme vu précédemment, il est assez simple d'obtenir un modèle de la classe à identifier sous réserve d'avoir une base de données satisfaisante. En contrepartie, ces méthodes peuvent avoir des difficultés à bien apprendre et donc ensuite classifier des objets présentant une grande variabilité. L'autre inconvénient vient des temps de calculs qui peuvent devenir conséquents quand l'espace de décision tend à devenir de dimension élevée. Pour pallier ces problèmes de temps, il existe plusieurs stratégies :

— la réduction du nombre de pas de parcours pour la résolution spatiale ou du nombre d'échelles mais cela peut poser des problèmes pour trouver des objets variables en taille.

— la réduction du nombre de paramètres dans le vecteur de caractéristiques [Papageorgiou et al., 1998] ce qui peut provoquer une augmentation des fausses détections du fait de la perte d'une partie de la pertinence du détecteur.

— l'utilisation de classifieurs en cascades [Viola and Jones, 2001, Chen and Chen, 2008]

— focaliser sur des caractéristiques connues à l'avance de l'objet : sa couleur [Sobottka and Pitas, 1996] ou dans une zone d'intérêt définie par l'utilisateur [Papageorgiou et al., 1998]

Nous verrons un peu plus en détails dans la partie suivante comment des ajouts de type top-down, c'est à dire guidés par le modèle peuvent être pertinents pour améliorer les résultats d'algorithmes bottom-up.

I.3.2 Méthodes top-down

Ces méthodes s'inspirent des mécanismes endogènes de la perception humaine. Elles utilisent un objectif fixé afin d'améliorer les performances et d'être moins sensibles aux distracteurs. Nous avons présenté précédemment des modèles principalement orientés bottom-up pour la reconnaissance d'objets : il s'agissait des modèles basés sur l'apparence. Nous allons ici de manière similaire présenter des modèles dont la conception est plutôt orientée top-down.

I.3.2.1 Modèles en parties

I.3.2.1.a Définitions

Une **partie** [1] est une division, une portion, un morceau, une fraction d'un tout plus complexe, ici l'objet. Une partie peut ainsi être par exemple un œil, une bouche, ou un nez par rapport à un objet "visage". Une partie est censée présenter un caractère moins invariant que l'entité (objet par exemple) qui la comprend.

Une **primitive** est le résultat renvoyé par un détecteur (un détecteur peut potentiellement retourner plusieurs primitives). Une primitive peut ensuite éventuellement être associée à une partie. La partie correspond donc au concept tandis que la primitive associée est la réalisation.

Par exemple, pour l'objet "carré aligné avec les axes", il est possible de construire des parties "bordHaut", "bordBas", "bordGauche", "bordDroit". Elles nécessiteront toutes le

1. En annexe, un glossaire rappellera les définitions importantes.

même type de primitives : "ligne". Par contre, pour chaque partie, s'il y a un seul carré présent, il y aura une seule association pertinente. Il existe potentiellement plus de primitives que de parties, les primitives ne sont donc pas toutes associées nécessairement à une partie. Il est également possible qu'une partie ne soit associée à aucune primitive : par exemple si la partie était occultée, le détecteur n'aura pas pu renvoyer la primitive correspondante.

Ces définitions étant posées, il est maintenant possible de s'intéresser plus précisément aux approches exploitant des modèles en parties.

I.3.2.1.b Principe des modèles en parties

À l'opposé des modèles basés sur l'apparence qui considèrent l'objet comme un tout à chercher dans son ensemble, les modèles en parties (**part-based models**) voient l'objet comme une collection de parties. La principale idée de ce découpage est que les parties doivent avoir une apparence plus uniforme et constante et donc être plus faciles à modéliser. Ces parties sont détectées individuellement et ensuite agglomérées ensemble à l'aide d'un modèle géométrique. Les parties peuvent avoir différentes positions les unes par rapport aux autres et le modèle géométrique permet de gérer ces variations. Une des raisons qui modifie l'agencement relatif des parties est le changement de point de vue de l'objet. Une autre raison peut venir de la nature déformable de l'objet, comme un piéton, qui admet donc des liaisons physiquement flexibles entre les différentes parties. Enfin, une troisième raison peut provenir de la variance intra-classe : différentes instances des objets d'une même classe peuvent avoir leurs parties à des positions différentes. Par exemple, dans la classe "chien", il est évident que la tête d'un dalmatien ou d'un teckel ne sera pas exactement à la même position par rapport au centre de gravité du chien. Même en considérant la classe "dalmatien", la position des pattes pourra varier selon que le dalmatien soit à l'arrêt ou en train de courir.

I.3.2.1.c Décomposition en Parties

Les premiers travaux dans la décomposition d'objets en parties se sont attachés à identifier des parties sémantiquement signifiantes. Ainsi une personne est composée d'un torse, d'une tête, de bras et de jambes. Les modèles définis ainsi ont l'avantage que les liens entre les différentes parties ont un sens sémantique physique et compréhensible. La détection d'une partie peut facilement être interprétée. L'avantage de tels modèles est qu'ils offrent un bon support pour la catégorisation : les mêmes primitives (mais pas les mêmes parties) peuvent être utilisées pour différents objets : une primitive *cercle* peut ainsi servir pour la détection de panneaux de signalisation de vitesse comme pour la détection de la roue d'un véhicule (sous certaines conditions d'angle). Le catalogue de primitives peut donc être restreint.

Par exemple, dans le travail de Leung [Leung et al., 1995], un ensemble de détecteurs est appliqué à l'image de test afin d'identifier les régions candidates aux caractéristiques faciales telles que les yeux, le nez, etc. La méthode proposée présente deux caractéristiques intéressantes : tout d'abord, elle permet de prendre en considération les parties manquantes de manière explicite et ensuite la combinatoire d'associations pour réussir l'appariement avec le modèle est réduite significativement grâce à l'introduction de contraintes sur la position relative des parties.

Les grammaires d'objets [Chen et al., 2006, Felzenszwalb and McAllester, 2011, Zhu and Mumford, 2006] sont d'autres applications intéressantes. Elles présentent l'avantage de pouvoir modéliser de manière élégante des variations intra-classe élevées, tout en conservant des parties avec une sémantique de haut niveau. Elles permettent également de gérer les occultations partielles. Dans [Lin et al., 2009], les auteurs ont appris une grammaire pour la détection de vélo, bien que les temps de calculs soient longs, ils parviennent à une bonne reconnaissance. De plus, ils montrent que selon la partie initialement choisie, la qualité et les temps pour parvenir au bon résultat varient. Ainsi il est montré qu'il est plus intéressant de commencer par la roue du vélo [2]. L'excessive combinatoire et les temps de calculs font partie des principaux inconvénients de ces méthodes.

Néanmoins, il faut usuellement définir manuellement les parties et elles peuvent être laborieuses à identifier ainsi qu'à annoter. De plus, leur détection est potentiellement un problème tout aussi complexe voire plus que la détection de l'objet au complet, ce qui nuit à leur intérêt premier : faciliter la reconnaissance de l'objet. Selon la pose, les parties peuvent s'occulter mutuellement ou être peu visibles en fonction de la position. Avec un modèle explicite (voir le § I.3.2.1) de l'objet, il est possible de gérer ces problèmes d'occultation mais cela peut complexifier le problème.

C'est pourquoi, d'autres méthodes choisissent de déterminer les parties de manière non supervisée ou faiblement supervisée [Bourdev and Malik, 2009, Si and Zhu, 2012] (et donc de manière plus automatique). Ces parties n'ont pas de traduction sémantique forte, par contre elles doivent être discriminantes et faciles à détecter dans une image. C'est le cas dans [Felzenszwalb et al., 2008] : un processus itératif est utilisé pour trouver un nombre donné de parties en initialisant les positions et tailles de parties réparties autour de l'objet. Puis les détecteurs sont entraînés pour chaque partie et enfin la position des parties est raffinée dans chaque image. La sortie de l'algorithme consiste alors en un modèle d'apparence pour chaque partie avec la position de chacune dans chaque image. D'autres exemples sont les **Implicit Shape Models** [Gall and Lempitsky, 2009, Lazebnik et al., 2006, Leibe et al., 2004] (ISMs). Leur principe est de définir de très nombreuses parties dont seul un sous ensemble sera visible dans chaque image. La détection de l'objet peut réussir malgré la non détection de certaines parties grâce à la densité des parties disponibles.

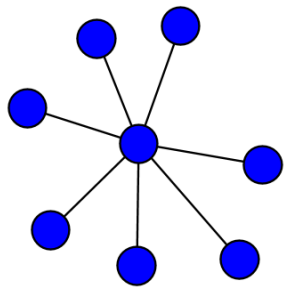
I.3.2.1.d Géométrie des parties

Quelle que soit la méthode de sélection des parties, il existe différentes méthodes géométriques pour les relier entre elles comme illustré dans la figure I.14.

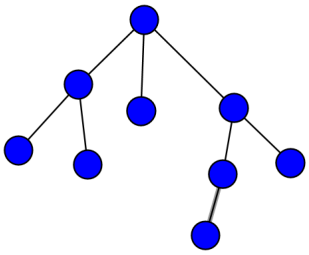
Le modèle le plus simple pour relier les parties entre elles est le **star model** [Leibe et al., 2004, Leibe et al., 2008, Gall and Lempitsky, 2009, Fergus et al., 2006, Felzenszwalb et al., 2008]. Chaque partie est connectée à un nœud central tout en restant indépendante des autres parties, ce qui permet de garder une complexité assez faible et autorise une **inférence** [3] rapide lors du processus de détection.

2. Notre système s'attachera d'ailleurs à privilégier un ordre dans la recherche des parties. Il l'inférera en fonction de l'état courant de la reconnaissance.

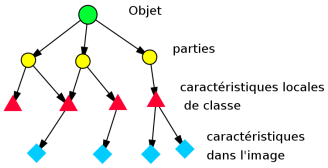
3. procédé par lequel il est possible d'estimer les caractéristiques de l'objet (et des autres parties) à partir des caractéristiques des parties détectées.



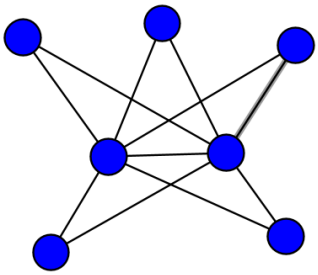
(a) Modèle en étoile : **Star model**



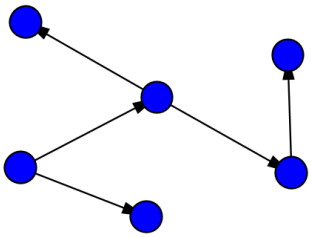
(b) Modèle en arbre : **Tree model**



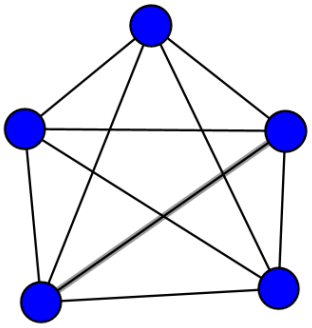
(c) Modèle hiérarchique : **Hierarchical model**



(d) Modèle $k-fan$ pour $k=2$: $k-fan$ **model**



(e) Modèle flexible épars pour $k=1$: **sparse flexible model**



(f) Modèle en constellation : **Constellation model**

FIGURE I.14 – Exemples de différents modèles géométriques inspirés de [Carneiro and Lowe, 2006]

Les modèles en arbre [Felzenszwalb and Huttenlocher, 2005] définissent une hiérarchie des parties où chacune a un seul parent et donc aucune connexion n'est autorisée entre les différentes branches. Cette structure permet aussi une inférence assez efficace. Les arbres ont aussi l'avantage d'être une structure assez naturelle pour des classes d'objets flexibles tels que le corps humain.

Beaucoup plus complexes, les modèles en constellation [Fergus et al., 2003, Fei-Fei et al., 2003, Cootes et al., 1995] imposent à toutes les parties d'être connectées à toutes les autres. Cela permet d'avoir une description beaucoup plus dense et autorise donc des interactions plus poussées entre les parties. La position de chaque partie dépend de la position de toutes les autres. Le principal inconvénient est qu'en conséquence, la complexité de l'apprentissage des liens et de l'inférence est exponentielle en fonction du nombre de parties.

Entre les deux extrêmes, il existe une multitude de modèles intermédiaires. Dans le model k-fan [Crandall et al., 2005], un groupe central de k parties entièrement connectées est entouré de parties avoisinantes connectées à ce groupe central mais pas aux autres parties (voir la figure I.14e avec $k=2$).

Les modèles hiérarchiques [Bouchard and Triggs, 2005] définissent un graphe orienté acyclique représentant l'objet, les parties de l'objet et les différentes caractéristiques permettant d'identifier les parties au niveau de la classe de l'objet et au niveau de l'image.

Dans les modèles épars flexibles [Carneiro and Lowe, 2006], le graphe des parties est défini de manière dynamique et la position de chaque partie dépend de la position de ses k voisines. Cela permet de représenter des objets flexibles tout en ajustant la complexité à l'aide de ce paramètre k.

I.3.2.1.e Détection finale de l'objet

Dans les modèles en parties, les parties individuelles sont extraites de l'image à l'aide soit de points d'intérêts, soit en appelant des détecteurs spécifiques (appris par exemple sur l'apparence de la partie comme dans le § I.3.1.2). Les résultats sont le plus souvent sous la forme de cartes des positions des candidats avec les scores pour chaque partie. L'étape suivante est alors de fusionner ces détections individuelles de parties en une détection d'objet géométriquement cohérente (en fonction du modèle précédemment choisi).

Dans les travaux de Felzenszwalb [Felzenszwalb et al., 2008], l'objet est détecté de manière grossière (la partie principale est l'objet entier) et ensuite la pose est précisée grâce aux autres parties (détecteurs plus fins). Chacune de ces autres parties utilise un histogramme de gradient orienté (HOG) comme représentation et un classifieur linéaire appris grâce à une SVM (voir le § I.3.1.2.d) capable de donner un score. La position optimale de la partie principale dépend alors à la fois de la position trouvée initialement (et du score associé à cette position) ainsi que des positions trouvées (et des scores) des autres parties. Un score pour chaque position est calculé. Il prend en compte le score de chaque partie détectée ainsi que le coût de déformation compte tenu de la position de la partie principale. Si la partie détectée est à une place cohérente par rapport à l'estimation de position de l'objet, le coût de déformation sera faible, sinon il sera plus élevé. La position avec le score le plus élevé indique la position la plus probable pour l'objet.

Dans les modèles de type **ISM** (Implicit Shape Models), un processus de vote est utilisé pour inférer la position de l'objet à partir de ces parties. Chaque partie vote pour une position de l'objet selon la distribution spatiale apprise du modèle. Le score d'un objet à une position et à une échelle donnée est alors la somme des votes de ses parties. Souvent, un lissage gaussien est appliqué pour permettre de petites déformations.

De manière générale, une fois le modèle géométrique choisi, les liens entre les parties sont appris et sont modélisés sous la forme d'une configuration optimale et de variations autorisées autour de cette configuration. Par exemple, pour un visage, il est possible de définir un écart moyen entre les yeux mais usuellement les yeux sont alignés horizontalement et donc la variation verticale d'un œil par rapport à l'autre sera faible (on ne peut pas avoir un œil au niveau de la bouche). La plupart du temps, ces variations sont simplement représentées par une distribution gaussienne avec une configuration initiale donnée par une valeur moyenne et les variations autorisées sont représentées grâce à une matrice de covariance [Felzenszwalb et al., 2008]. Des mixtures de gaussiennes peuvent également être utilisées [Gall and Lempitsky, 2009, Leibe et al., 2004, Leibe et al., 2008]. L'avantage de ce type de représentations est qu'il permet d'obtenir la position globale en prenant en compte les positions des différentes parties détectées tout en gardant une cohérence géométrique forte. Il autorise aussi les variations des positions des parties ce qui est appréciable pour des objets déformables sans pour autant nuire à la détection des objets rigides : les variations autorisées seront plus ou moins élevées selon les cas. La détection des parties entraîne donc bien une estimation de la position de l'objet.

Nous avons donc vu des modèles presque purement top-down, nous allons maintenant présenter comment des aspects top-down peuvent améliorer des méthodes originellement conçues de manière plutôt bottom-up. C'est le cas dans les cartes de saillance où associer les deux modalités (top-down et bottom-up) permet de mieux imiter la perception humaine.

I.3.2.2 Modification des cartes de saillance

Les cartes de saillance définies dans le § I.3.1.1 étaient majoritairement orientées bottom-up dans leur conception. Cependant, nous avons montré précédemment que la tâche avait un impact important sur l'attention humaine [Yarbus, 1967] en particulier pour des actions de recherche volontaire. En fait, Henderson [Henderson et al., 2007] a montré que les informations issues des processus top-down dominaient lors des phases de recherche dans des scènes réelles. D'autres ont mis en exergue l'importance du contexte pour la recherche [Oliva et al., 2003, Torralba et al., 2006]. Pour imiter correctement l'attention humaine, fusionner des processus top-down et des processus bottom-up est intéressant.

Le contexte de la scène est important pour accélérer la recherche et la reconnaissance : si par exemple, l'objet cherché est une personne, les "objets" au sol seront examinés prioritairement par rapport aux "objets" dans le ciel (il est plus probable d'avoir un piéton qu'un parachutiste). Ces informations liées au contexte peuvent être ajoutées aux modèles. Ainsi, dans [Torralba et al., 2006], les auteurs combinent des éléments bas niveau avec le contexte de la scène pour guider la recherche. Les aires avec une forte saillance qui se trouvent dans une région globale donnée ont un poids plus important que celles tombant en dehors de la zone. Ce modèle de recherche contextuel donne de meilleurs résultats que les modèles purement bottom-up pour la prédiction des fixations humaines lors d'une tâche de recherche. Ces recherches ont été poursuivies par [Ehinger et al., 2009].

Dans le même esprit, les auteurs de [Zhang et al., 2008] et de [Kanan et al., 2009] ont créé le modèle SUN (Saliency Using Natural statistics) qui combine des informations top-down et bottom-up pour prédire les mouvements des yeux lors de tâches de recherche dans des images de scènes réelles. Cependant contrairement au modèle précédent, la méthode SUN implémente les caractéristiques de la cible comme composantes top-down. Encore une fois, cette méthode surpasse les méthodes purement bottom-up pour la prédiction des fixations humaines dans des images réelles. Ces deux méthodes montrent donc la pertinence d'utiliser les deux composantes (bottom-up et top-down) afin d'augmenter significativement leurs capacités à prédire où regarderait un humain.

Dans [Goferman et al., 2012], les auteurs ont développé un algorithme dont le but est de détecter dans des images les régions qui représentent le mieux la scène et pas seulement les objets les plus saillants. En plus d'inclure des caractéristiques bas niveau telles que le contraste ou la couleur, ils sont capables de supprimer les objets récurrents dans l'image et ajoutent la notion de forme visuelle avec plusieurs centres de gravité ainsi que des détecteurs de visages.

Une autre manière d'intégrer des composantes top-down est de moduler les poids des différentes caractéristiques bas niveau en fonction de la tâche à accomplir. C'est une idée proposée originellement par Wolfe [Wolfe et al., 1989]. Par exemple, si l'objectif est de trouver une bouteille verte alignée verticalement, le modèle va augmenter les poids pour les cartes correspondant aux caractéristiques "vert" et "orientation verticale" afin de

permettre à ces caractéristiques d'être considérées comme plus saillantes. Dans la carte de saillance ainsi formée, toutes les régions dont les caractéristiques sont similaires à l'objet cible deviennent plus saillantes et sont plus propices à attirer l'attention.

Ce type d'approches est utilisé dans [Elazary and Itti, 2010, Gao et al., 2008, Navalpakkam and Itti, 2007] par exemple. Elezary et Itti [Elazary and Itti, 2010] proposent un modèle appelé *SalBayes* qui associe le principe de saillance et un modèle bayésien. Ce modèle apprend les probabilités que l'apparence d'un objet soit dans un ensemble de valeurs pour des cartes de caractéristiques données. Lors d'une tâche de recherche, le modèle influence les différentes cartes de caractéristiques en calculant la probabilité que l'objet cherché puisse être trouvé dans chaque carte de caractéristiques. Les régions avec la plus grande probabilité sont alors examinées en priorité.

Marchesotti et al [Marchesotti et al., 2009], quant à eux, utilisent le contexte en proposant un modèle de carte de saillance qui part du principe que les images ayant une apparence globale similaire doivent partager une saillance semblable. En admettant qu'une large base d'images annotées est disponible, ils récupèrent les images les plus similaires à l'image cible, construisent un classifieur simple et utilisent le classifieur ainsi créé pour générer des cartes de saillance. Leur principale application est l'extraction de sous images pertinentes.

Semblablement, dans [Gao et al., 2008, Gao and Vasconcelos, 2004], les auteurs proposent un modèle unifié pour traiter à la fois la saillance bottom-up et top-down comme un problème de classification. Ils ont appliqué ce modèle pour la détection d'objets [Gao and Vasconcelos, 2005]. Dans ces travaux, un ensemble de caractéristiques est sélectionné afin de permettre de séparer la classe recherchée des autres classes. La saillance est alors définie grâce à la somme pondérée des caractéristiques considérées comme saillantes pour la classe donnée.

Une troisième manière d'ajouter une composante top-down aux modèles est d'incorporer l'utilisation de détecteurs d'objets. Dans [Cerf et al., 2008, Cerf et al., 2009b, Cerf et al., 2009a], les auteurs montrent que les visages et les textes attirent très fortement l'attention et qu'il était difficile de les ignorer. Ils ont retravaillé le modèle d'Itti and Koch [Itti et al., 1998] en ajoutant une carte qui intègre la position des visages et des écritures et ont montré que cela améliorait les performances du modèle pour prédire où se fixe le regard dans des images naturelles. De plus, Einhauser et ses collègues ont montré dans [Einhauser et al., 2008] que les fixations étaient mieux prédites en déterminant la position des objets qu'en calculant une saillance de bas niveau.

Nous venons de voir une manière de moduler les cartes de saillance afin de prendre également en compte des composantes top-down. Il peut s'agir de prendre en compte le contexte, de moduler le poids des différentes cartes de caractéristiques ou encore d'ajouter l'influence d'objets haut niveau tels que des visages vers lesquels le regard humain est attiré. L'objectif peut être d'imiter au mieux l'attention humaine en prenant en compte au moins partiellement des mécanismes top-down qui peuvent s'apparenter dans l'idée (le fonctionnement réel étant beaucoup plus mystérieux) aux mécanismes endogènes de l'attention ou de faciliter la détection d'une cible précise en la rendant plus saillante dans les images.

I.3.3 Bilan des méthodes de perception artificielle

Nous avons vu deux grands types d'approches de perception artificielle : les méthodes dites bottom-up et les méthodes top-down. Ces méthodes ont été développées afin d'imiter les mécanismes de l'attention humaine que sont les mécanismes endogènes et exogènes.

Les méthodes bottom-up sont dans l'ensemble faciles à mettre en œuvre et ne nécessitent pas une *intelligence* de l'algorithme. Elle peuvent être très performantes à la fois en terme de précision et de rapidité. Elles permettent également de gérer facilement l'imprévu puisqu'elles n'ont aucun a priori et sont donc ouvertes à toutes les possibilités. En contrepartie, selon les objets et les scènes, elles peuvent se laisser distraire et être plus sensibles au bruit. Enfin, les modèles fondés sur l'apparence permettent difficilement de gérer une grande complexité intra-classe et les détecteurs doivent souvent être particulièrement robustes. Les occultations ne sont pas ou peu gérées : la plupart du temps, il s'agit juste de constater la plus ou moins grande sensibilité des algorithmes à ces problèmes.

À l'inverse, les méthodes top-down sont souvent plus complexes à mettre en œuvre car elles nécessitent une modélisation préalable. Néanmoins elles autorisent la prise en compte des connaissances a priori sur les objets. Le découpage de l'objet en parties est un atout pour la gestion des occultations en acceptant la non détection d'une ou plusieurs parties sans pénaliser l'algorithme. Les liens connus entre les parties permettent de raffiner la pose et facilitent les regroupements (seules les parties compatibles avec celles déjà trouvées seront traitées plus en avant). Néanmoins, ce découpage en parties peut être problématique car prises individuellement de telles parties sont parfois plus délicates à reconnaître que l'objet dans sa totalité. Enfin, ce sont souvent des algorithmes gourmands en temps de calculs.

Comme pour les mécanismes attentionnels humains, bien que présentés séparément, il existe des interactions entre les deux. C'est le cas par exemple pour les cartes de saillance pour lesquelles nous avons vu qu'il était possible d'augmenter la qualité (par rapport à l'imitation du regard humain) en ajoutant des composantes top-down. C'est également le cas de la modélisation en parties : chaque partie peut être apprise avec un modèle basé sur l'apparence et être détectée de manière purement bottom-up même si pour des raisons de temps de calculs, il est plus intéressant de limiter la recherche à des zones pertinentes.

I.4 Bilan sur la perception humaine et artificielle

La perception humaine est un moyen très efficace de se rendre compte du monde extérieur qui nous entoure. Elle est capable de s'adapter à des situations très variables et nous permet d'accomplir une multitude de tâches du quotidien que ce soit pour nous localiser dans l'espace ou encore pour reconnaître des objets, interpréter des scènes et interagir en conséquence.

C'est pourquoi des algorithmes ont cherché à imiter cette perception humaine. Celle-ci dispose de deux mécanismes principaux permettant de focaliser l'attention sur une faible sous parties des stimuli en provenance du monde extérieur. Les premiers, les mécanismes exogènes sont indépendants de notre volonté et sont attirés par des éléments saillants (qui sautent "aux yeux"). Ils n'ont aucun a priori. Ils permettent de gérer certains imprévus en fournissant justement des informations auxquelles on ne s'attendait pas comme pour un

changement de signalisation routière. Les seconds, les mécanismes endogènes sont au contraire guidés par l'objectif, ils nécessitent d'avoir des connaissances sur la tâche de perception à réaliser afin de fournir des solutions efficaces. Ils essaient de sélectionner les informations pertinentes par rapport au but fixé et sont peu sensibles aux distracteurs.

Pour imiter les mécanismes exogènes, des algorithmes dit bottom-up ont été développés. Il existe notamment de nombreux algorithmes permettant de créer des cartes de saillance [Itti et al., 1998, Achanta et al., 2009, Navalpakkam and Itti, 2006, Le Meur et al., 2005, Urban et al., 2010] qui mettent en exergue les objets les plus saillants dans une image donnée. La plupart se sont fondés sur la *théorie d'intégration des caractéristiques* de Treisman and Gelade [Treisman and Gelade, 1980] et du modèle de recherche orientée de Wolf et al [Wolfe et al., 1989] qui propose justement un modèle biologiquement plausible pour l'attention visuelle. Ces cartes de saillance permettent de prédire où un être humain regarderait "naturellement" dans une image, c'est-à-dire lors d'une exploration libre et non guidée par un objectif donné. En plus d'essayer de modéliser (et donc de comprendre) le fonctionnement de la perception humaine, elles ont diverses applications telle que l'aide à la navigation [Siagian and Itti, 2009], à la segmentation [Valenti et al., 2009], à la compression [Chenlei Guo and Liming Zhang, 2010] ou encore à la simplification d'images [Grabli et al., 2004].

La plupart du temps, ces cartes de saillance prennent essentiellement en compte des éléments de bas niveau comme des différences de couleurs ou de gradients. Néanmoins, l'être humain dispose aussi de détecteurs automatiques de haut niveau : par exemple il est souvent attiré par des visages dans une image [Cerf et al., 2008]. C'est pourquoi, les méthodes bottom-up ont aussi été utilisées dans le but de reconnaître des objets complets de manière rapide. Ce sont les techniques basées sur l'apparence [Viola and Jones, 2004, Bourdev and Brandt, 2005, Sankari and Adeli, 2011, Dollár et al., 2012]. Ces techniques ont besoin le plus souvent d'avoir appris un modèle à l'aide d'une base d'apprentissage [Schölkopf and Smola, 2002a] qui nécessite en général un grand nombre d'exemples [Jain et al., 2000] la rendant fastidieuse à construire. Une fois le modèle appris, il suffit d'appeler un classifieur tel que les SVM [Cristianini and Shawe-Taylor, 2000] ou encore les réseaux de neurones [Krizhevsky et al., 2012] dans toute l'image. L'utilisation est donc particulièrement simple et se prête bien aux calculs force brute des ordinateurs d'aujourd'hui. Elles peuvent donc se révéler très rapides et performantes selon le contexte et les objets à détecter.

Les méthodes top-down sont quant à elles inspirées des mécanismes endogènes. Afin d'orienter la recherche, des modèles en parties [Leung et al., 1995, Felzenszwalb and McAllester, 2011, Fergus et al., 2003] sont créés : au lieu de considérer comme dans les techniques basées sur l'apparence l'objet comme un tout, il est vu comme une agrégation de différentes parties. Ces parties peuvent soit avoir été définies manuellement (avec une sémantique haut-niveau) [Zhu and Mumford, 2006] ou automatiquement [Si and Zhu, 2012]. Une fois les parties définies, l'objectif est de trouver ces différentes parties et grâce aux liens existant entre elles (et entre elles et l'objet) de retrouver l'objet. Ces liens peuvent être modélisés de différentes manières : il existe des modèles en étoile [Leibe et al., 2004], en constellation [Fei-Fei et al., 2003] ou encore hiérarchiques [Bouchard and Triggs, 2005]. Un inconvénient de ces méthodes est que la modélisation de l'objet et l'inférence nécessaire pour estimer l'objet à partir de ses parties peuvent être complexes et peu efficaces. En contrepartie, cette modélisation permet une gestion explicite des occultations comme des

parties manquantes et autorise une focalisation spatiale (une fois un œil gauche trouvé, l'œil droit doit être à proximité) et sur les caractéristiques (si le premier œil était vert, sauf cas particulier, le deuxième sera vert aussi). Cette focalisation sur les caractéristiques correspond typiquement aux mécanismes endogènes qui ne s'intéressent qu'aux éléments pertinents par rapport au modèle. La focalisation spatiale évoque dans la vision humaine les déplacements oculaires qui cherchent à placer la fovéa sur la zone intéressante ce qui permet des traitements complexes. La focalisation spatiale provoque un système similaire : les détections suivantes seront réalisées uniquement dans des zones restreintes.

Enfin, si les mécanismes endogènes et exogènes interagissent chez l'être humain, c'est aussi le cas pour les méthodes bottom-up et top-down. Nous avons ainsi présenté des améliorations des cartes de saillance à l'aide de procédés top-down. Ainsi, comme dans les méthodes top-down, des traitements en cascade [Viola and Jones, 2001] permettent de ne pas traiter entièrement toutes les zones de l'image et correspondent donc à une focalisation spatiale. Il peut également y avoir une focalisation en fonction des caractéristiques soit en se concentrant uniquement sur les zones particulières [Papageorgiou et al., 1998, Sobottka and Pitas, 1996], soit en intégrant les caractéristiques de l'objet cherché [Elazary and Itti, 2010, Gao and Vasconcelos, 2005] pour définir ce qui est saillant.

Malgré tout, la perception artificielle reste un problème complexe qui n'a pas été entièrement résolu et dont les capacités demeurent très en deçà de celles de la perception humaine. Si les méthodes bottom-up peuvent être rapides et efficaces, elles ne permettent pas la gestion des occultations et se contentent le plus souvent uniquement d'indiquer quel est leur degré de résistance à ces occultations. De plus, dans le cas d'objets très déformables ou de classe d'objets avec une grande variabilité, l'apparence est trop changeante pour pouvoir être correctement définie comme un tout et donc ces méthodes ne sont pas très adaptées. À l'inverse, les méthodes top-down permettent une modélisation complète de l'objet. Elles peuvent donc gérer explicitement les occultations ainsi que les parties manquantes. Néanmoins, ces méthodes sont souvent assez lentes et obtiennent fréquemment des résultats en deçà des méthodes bottom-up.

Chapitre II

Stratégie de Perception Artificielle

« *La réalité est une chose mystérieuse et fluctuante, car la perception que nous en avons ne reste jamais la même.* »

Joe Tan

Sommaire

II.1 Objectifs

La perception est notre moyen d'avoir accès au monde extérieur et permet la construction d'une représentation à partir des stimuli du monde extérieur. L'objectif de cette thèse est de définir une stratégie de perception qui se rapproche de la perception humaine. Nous cherchons donc à définir un ensemble d'actions cohérentes permettant d'aboutir *habilement* à la tâche demandée.

Nous cherchons à construire un algorithme générique permettant potentiellement de réaliser différents types de perception comme la perception auditive et visuelle. En outre, comme la perception humaine, l'objectif visé par l'algorithme doit pouvoir inclure des domaines variables. Ainsi, nous souhaitons à l'aide du même algorithme pouvoir aussi bien traiter des problèmes de reconnaissance d'objets dans une image que d'être capable de localiser un véhicule dans le monde en utilisant différents capteurs (comme des télémètres laser, des récepteurs gps, etc) et des cartes de l'environnement. Il faut donc que l'algorithme soit capable de gérer différents capteurs et détecteurs et d'en fusionner l'information dans une représentation cohérente. De plus, l'algorithme devra être modulable afin de pouvoir ajouter (ou supprimer) facilement des capteurs et des détecteurs.

Pour la suite, la représentation cohérente que nous souhaitons construire en fonction d'un objectif donné sera appelée *objet*. Compte tenu des exigences de généricité mentionnées ci-dessus, il faut donc être capable de modéliser des caractéristiques de *l'objet* très variables. La représentation de l'objet devra donc permettre d'inclure des éléments liés à l'aspect (la texture, la forme, la couleur) aussi bien que des informations de position (orientation, localisation dans l'espace) ou encore des caractéristiques liées au toucher (granuleux, mou, etc) ou à l'ouïe (aigu, grave). Tout ce qui peut être pertinent pour identifier l'objet et qui fait donc partie de sa représentation souhaitée doit pouvoir être intégré.

Dans le chapitre précédent, nous avons vu que la perception humaine en plus d'être très générique était également très performante. C'est pourquoi nous souhaitons nous en inspirer pour construire notre algorithme. Plusieurs points nous semblent particulièrement intéressants. Tout d'abord, les êtres humains disposent de deux mécanismes d'attention : des mécanismes exogènes et des mécanismes endogènes qui leur permettent de ne sélectionner qu'une petite partie des informations en provenance du monde extérieur. Avoir dans notre système de perception artificielle, un tel mécanisme de l'attention nous parait donc indispensable. De plus, nous avons vu qu'intégrer de tels mécanismes amélioraient souvent le processus de détection [Viola and Jones, 2001, Felzenszwalb et al., 2010, Aynaud, 2015]. Nous retenons plus particulièrement les mécanismes endogènes qui servent entre autres à éliminer les distracteurs lorsqu'un objectif est fixé, ce qui est notre cas.

Concernant les mécanismes endogènes, nous notons de plus que ceux-ci, chez l'être humain, exigent une représentation préexistente de la tâche à accomplir. Il nous parait donc intéressant de chercher à prendre en compte des connaissances a priori dans notre système de perception et ce d'autant plus que de telles connaissances sont de plus en plus accessibles. Il existe ainsi de plus en plus de modèles 3D d'objets (comme des véhicules par exemple) qui peuvent servir pour leur reconnaissance. Le même principe peut être utilisé pour la localisation de robots : nous verrons que dans ce cas la connaissance préalable sera une carte de l'environnement.

Un autre point intéressant de la perception humaine et des mécanismes endogènes est la concentration de l'attention sur une zone restreinte (la focalisation d'attention). Ainsi nous avons vu que pour la vision humaine, seule la fovéa permettait des traitements précis de l'information et que le reste de la rétine ne fournissait que des informations plus grossières. Ce principe nous semble transposable pour notre perception artificielle. Par exemple, dans une image, il est possible d'imaginer des traitements grossiers permettant d'éliminer rapidement les zones sans intérêt (par rapport à la tâche donnée) et de focaliser uniquement sur un ensemble de sous images plus petites que l'image d'origine. Les traitements plus lourds seront alors uniquement lancés dans ces zones restreintes. De manière plus générale, seule une partie des données capteurs fournies serait examinée plus en détail.

Une autre capacité de la perception humaine qu'il nous semble judicieux d'imiter, est celle permettant de gérer de manière pertinente plusieurs capteurs. Ainsi, un être humain est capable de prendre en compte les capacités et les limitations de ses différents organes de sens. Par exemple, dans le noir complet, il se fiera peu à ses yeux et préférera avancer en tâtonnant. Il est donc capable de faire une sélection en fonction des circonstances du capteur le plus adapté et peut gérer les imperfections de ses capteurs. Nous souhaitons donc que l'approche proposée soit capable d'avoir une sélection similaire. Nous voulons également étendre ce principe non plus aux seuls capteurs mais également aux détecteurs : en admettant connaître les forces et les faiblesses de nos détecteurs, nous sélectionnerons au mieux en fonction de notre objectif et des circonstances, un capteur et un détecteur les plus adaptés.

Enfin, un dernier point que nous estimons pertinent à imiter de l'être humain est sa capacité à probabiliser et à se remettre en question. Par exemple, si une personne cherche son chemin et qu'elle hésite à une intersection entre deux routes, elle en choisira une tout en ayant conscience de s'être potentiellement trompée. Si en poursuivant sur la route choisie, elle repère de nouveaux indicateurs qu'elle espérait trouver (comme un panneau de direction par exemple), elle sera plus confiante dans le choix réalisé. Si, à l'inverse, elle ne repère rien, elle pensera avoir fait le mauvais choix et reviendra sur ses pas. Nous souhaitons donc intégrer cette notion de "confiance" dans notre algorithme afin de pouvoir similairement revenir sur un choix effectué et d'être en mesure à chaque instant d'indiquer à quel point l'algorithme estime être sur la bonne voie par rapport à la tâche qu'il essaie d'accomplir. Nous voulons également que l'algorithme puisse fournir une précision à son estimation (est-on précis à 10 mètres, à 100 mètres ou à 1 cm par exemple pour le cas d'un problème de localisation).

Ainsi, il est possible de résumer les différentes caractéristiques que nous souhaitons avoir pour notre algorithme, il doit être :

— générique
— multicapteurs
— modulaire
— capable d'utiliser les connaissances a priori
— capable de focaliser : à la fois sur les zones pertinentes (focalisation *spatiale*) mais aussi sur les caractéristiques recherchées pour l'objet (focalisation *de caractéristiques*).

— capable de sélectionner les données pertinentes en fonction de l'objectif à atteindre
— capable de donner une précision à son estimation courante
— en mesure d'expliciter la confiance qu'il a dans son estimation
— capable de revenir en arrière si nécessaire.

II.2 Choix Stratégiques

Dans le chapitre précédent, nous avons vu deux grandes catégories de méthodes inspirées des mécanismes attentionnels de la perception humaine. Nous allons ici rapidement revenir sur les avantages et les inconvénients de ces deux types de méthodes afin de déterminer celle qui nous conviendrait le mieux compte tenu des objectifs donnés précédemment.

Les méthodes bottom-up [Itti et al., 1998, Achanta et al., 2009, Navalpakkam and Itti, 2006, Le Meur et al., 2005, Urban et al., 2011] exploitent directement les données capteurs. Elles sont actuellement très utilisées car elles sont faciles à mettre en œuvre, se prêtent bien aux calculs force brute des ordinateurs d'aujourd'hui. Elles ont l'avantage de pouvoir détecter des événements (ou des objets) qu'il n'était pas possible d'anticiper. En contrepartie, elles ne peuvent pas prévoir l'avenir et souvent ne prennent pas en compte les connaissances disponibles sur l'objet. Si selon les applications, ces méthodes sont particulièrement rapides et performantes [Viola and Jones, 2004], elles peuvent également engendrer des coûts de calculs conséquents en particulier dans les système multi capteurs car toutes les données doivent être utilisées et analysées. Pour les méthodes basées sur l'apparence [Bourdev and Brandt, 2005, Sankari and Adeli, 2011, Dollár et al., 2012] demeure le problème de la construction de la base d'apprentissage [Schölkopf and Smola, 2002a]. La qualité de la base d'apprentissage peut grandement influencer le résultat de la détection. Enfin, comme ces méthodes s'appuient directement sur les données, leur gestion de l'occultation se réduit le plus souvent à estimer leur robustesse à celle-ci. Ainsi les occultations sont subies : rien n'est fait pour les prendre en compte. Ces méthodes bottom-up ne gèrent pas ou difficilement des classes d'objets présentant une trop grande variabilité.

Les méthodes top-down [Leung et al., 1995, Felzenszwalb and McAllester, 2011, Fergus et al., 2003], quant à elles, sont guidées par l'objectif à atteindre. Elles sont, le plus souvent, moins sensibles aux distracteurs notamment grâce à la prise en compte des connaissances sur la tâche à accomplir. En contrepartie, elles ne permettent pas la gestion des imprévus. Avec les modèles en parties [Bouchard and Triggs, 2005, Fei-Fei et al., 2003], il est possible de gérer explicitement les non détections qu'elles soient liées à des occultations ou à la non existence de certaines parties. Néanmoins, la définition de ces parties n'est pas toujours aisée. Si l'utilisateur définit lui-même les parties intéressantes, il risque potentiellement d'en choisir des peu pertinentes ou difficilement détectables. Mais si elles sont construites automatiquement, il est plus complexe de gérer les occultations. Si ce découpage en parties peut permettre une focalisation à la fois *spatiale* et *de caractéristiques*, ces méthodes restent malgré tout le plus souvent assez lentes à cause entre autres de l'inférence nécessaire pour transformer la détection des parties en détection de l'objet. De plus, la construction d'un modèle complet de l'objet modélisant correctement toutes les connaissances a priori n'est pas toujours évidente.

Le tableau II.1 récapitule les principaux avantages et inconvénients de ces deux types de méthodes.

TABLEAU II.1 – Avantages et Inconvénients des méthodes top-down et bottom-up.

	Avantages	Inconvénients
top-down	• prise en compte des connaissances a priori • gestion des occultations • gestion des parties manquantes • détection d'objets très déformables • focalisation *spatiale* • focalisation *de caractéristiques*	• pas de gestion de l'imprévu • difficulté de modélisation problème du découpage en parties • complexité de l'inférence • temps de détection
bottom-up	• gestion des imprévus • facilité d'utilisation • rapidité de fonctionnement selon les applications • existence de base d'apprentissages déjà annotées pour certaines classes d'objets	• non utilisation des connaissances a priori • pas de gestion explicite des occultations • utilisation de toutes les données : temps de calcul potentiellement élevé (multi capteurs) • nécessité de création d'une base d'apprentissage • difficulté pour la détection de classe d'objets très variables en apparence.

Comme nous souhaitons développer une application qui puisse utiliser les connaissances a priori et qui permette de sélectionner les données pertinentes, les méthodes de type top-down paraissent plus appropriées. En effet, il est judicieux lorsqu'un piéton est recherché par exemple, d'admettre que celui-ci doit marcher sur le sol et qu'en conséquence, la zone du ciel peut être délaissée sans perte d'information par rapport au but recherché. Souvent, les objets à détecter sont des objets pour lesquels un ensemble de caractéristiques admissibles peut être défini. Ainsi, si le feuillage d'un arbre se prête mal à ce type de modélisation, les objets fabriqués par l'homme comme les immeubles, les poteaux, les voitures ont souvent au contraire des dimensions et des formes assez standardisées. Même parmi les "objets" plus naturels, comme un chat ou un chien, il peut être opportun de profiter d'éléments communs comme la taille (un chat ne fait pas deux mètres de haut), ou encore la proportion entre les différents éléments (la tête ne sera jamais deux fois plus grosse que le corps).

Parmi les méthodes top-down, nous avons vu les méthodes impliquant l'ajout d'une ou plusieurs composantes top-down ou les méthodes nécessitant un découpage en parties.

Les deux peuvent permettre d'avoir une focalisation *de caractéristiques* : en ajoutant une composante liée à l'aspect de l'objet cherché pour les premières [Wolfe et al., 1989, Gao et al., 2008] ou grâce aux liens entre les parties pour les secondes. Les deux peuvent aussi offrir une focalisation *spatiale* en se concentrant sur des zones spécifiques de l'image. Néanmoins, la gestion des occultations est plus simple à faire avec des modèles en parties [Leung et al., 1995]. C'est pourquoi, parmi les méthodes top-down, nous avons sélectionné les méthodes impliquant un découpage de l'objet en parties. Bien que présentée dans le cadre de la reconnaissance d'objets et donc avec des parties correspondant à des morceaux visuels d'objets, on prendra ici un sens plus large de ce mot afin de pouvoir inclure toute caractéristique de l'objet qu'elle soit visuelle, sonore ou autre. En ayant une modélisation explicite de l'objet, il est envisageable de gérer de manière explicite les problèmes d'occultation des parties ou de manière plus générale, les problèmes liés à l'environnement gênant pour la détection d'une ou plusieurs parties. Ainsi si la partie correspond à un signal électromagnétique donné, la présence d'équipements électriques à proximité pourra être traitée comme une *occultation* qui nuit à la reconnaissance. De plus, cette modélisation peut éventuellement autoriser d'avoir des parties optionnelles : c'est le cas par exemple dans les grammaire d'objets [Chen et al., 2006, Felzenszwalb and McAllester, 2011, Zhu and Mumford, 2006] où les chiffres d'une horloge sont catégorisés en chiffres arabes, chiffres romans ou chiffres non apparents.

Pour le découpage de l'objet en parties, deux grands choix s'offrent à nous : soit déterminer de manière automatique les différentes parties comme dans [Bourdev and Malik, 2009, Si and Zhu, 2012] ou bien avoir un découpage manuel en parties. L'avantage du premier type de méthode est que les parties définies seront pertinentes et détectables. Laisser à l'utilisateur le soin de définir les différentes parties qui composeront l'objet peut être source de difficultés en permettant par exemple d'avoir des parties réellement complexes à détecter (parfois même plus que l'objet dans sa globalité). Cependant, cela laisse une plus grande flexibilité pour leur définition et permet de pouvoir rajouter au fur et à mesure des parties en fonction des besoins : puisque chaque partie a une existence propre et une définition nette au moins pour l'utilisateur, il est possible de la relier à l'objet. L'autre avantage d'avoir des parties connues et identifiables est la possibilité de rajouter des détecteurs. Ainsi, par exemple, si une partie de l'objet correspond à une ligne dans une image, différents détecteurs peuvent être utilisés : un détecteur fondé par exemple sur l'extraction de contours suivie d'un RANSAC ou un autre qui s'appuierait plutôt sur la méthode de Hough.

Enfin, il n'est pas impossible d'imaginer être ultérieurement capable de gérer de manière automatique l'ajout ou la suppression de parties. Par exemple dans le cadre d'une application de localisation, il faudrait idéalement mettre à jour une carte préexistante fournie : certains arbres ont pu être enlevés tandis que de nouveaux bâtiments seraient apparus. Les parties seraient toujours sémantiquement significatives mais il devrait être possible de les supprimer ou d'en rajouter en fonction des résultats des détections. Comme nous souhaitons avoir un algorithme modulaire, nous opterons donc pour un découpage manuel des parties. Un des objectifs de l'algorithme que nous proposerons sera alors de choisir parmi tous les détecteurs disponibles pour chaque partie les plus appropriés selon les circonstances.

Pour faire ce choix de détecteurs, il est nécessaire de bien les connaître, c'est à dire d'être capable d'estimer le temps qu'ils mettront à répondre, quelle est la qualité de la

détection rendue que ce soit en terme de précision (pixellique, métrique ou autre selon le capteur considéré) ou en terme de fiabilité. Pour la fiabilité, il existe deux indicateurs dont la terminologie est empruntée à la reconnaissance d'objets dans des images mais qui peut s'appliquer pour d'autres modalités capteur : le taux de faux positifs ainsi que le taux de faux négatifs. Le premier désigne le pourcentage de fois où le détecteur renvoie *oui* par rapport à ce qu'il recherchait (il indique qu'il a trouvé quelque chose semblant correspondre) alors que c'est faux. Dans une image, c'est par exemple, le cas du poteau avec un panneau de limitation placée à une hauteur critique qui est détecté à tort comme l'item "piéton". Le taux de faux négatifs renvoie quant à lui au pourcentage de fois où le détecteur ne renvoie rien alors que l'objet (ou la partie de l'objet) qu'il recherche est présent.

Nos travaux ont été initiés par [Aufrère et al., 2000] et s'appuient également sur [Chapuis et al., 2008, Trujillo Morales, 2007]. Ces travaux proposent déjà de gérer la variabilité des objets à détecter ainsi que les imprécisions liées aux détecteurs. Ils ont également été adaptés avec succès pour des problèmes de localisation [Tessier et al., 2009, Aynaud et al., 2014]. Néanmoins, nous nous démarquons en proposant une approche plus flexible et fondée sur des parties identifiables contrairement à [Trujillo Morales, 2007] où le choix est fait de sélectionner les caractéristiques les plus adéquates parmi un ensemble fixé. De plus, nous proposons de gérer de manière explicite les problèmes d'occultation d'une ou plusieurs parties. Enfin, nous apporterons une simplification dans la modélisation des événements pris en compte (occultation, fiabilité des détecteurs ...) pour les erreurs de détection ce qui permet de complexifier le modèle en fonction des besoins.

Le schéma II.1 montre à quel niveau nous nous plaçons. Nous partons d'un objectif haut-niveau qui va aller chercher via un processus top-down (flèche violette pleine) des données moyen niveau (les parties de notre objet) obtenues à l'aide d'un procédé bottom-up (flèche jaune pleine) : c'est l'utilisation de nos détecteurs qui extraient les parties des données capteurs. Nous nous intéressons donc essentiellement aux aspects haut niveau et nous prendrons en entrée de notre système des données moyen niveau issues de détecteurs de bas niveau dont nous ne nous occuperons pas beaucoup dans ce travail de thèse.

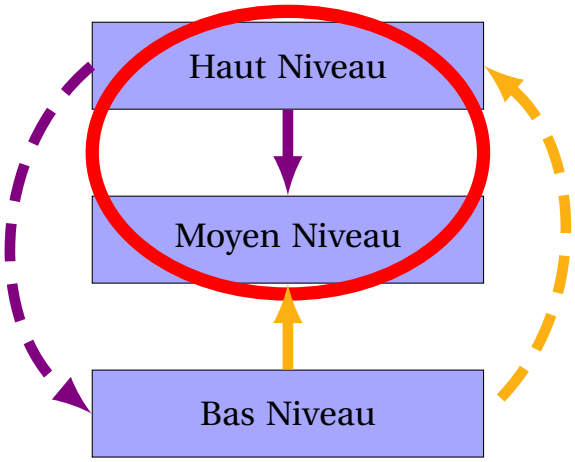


FIGURE II.1 – Schéma représentant le positionnement de l'algorithme par rapport au bas, moyen et haut niveau. Les flèches pleines représentent ce qu'il est prévu de gérer par l'algorithme, les flèches en pointillés sont les ajouts idéaux à faire. En violet, ce sont les aspects top-down qui sont représentés et en jaune, les aspects bottom-up.

L'idéal serait d'avoir un processus réalisant également les tâches montrées par des flèches en pointillé. D'une part, il serait intéressant d'avoir un processus purement bottom-up permettant de gérer les imprévus (comme signaler un obstacle non prévisible). D'autre part, lorsque les détecteurs échouent à détecter une partie, il pourrait être utile lorsqu'on sait que l'objet et plus particulièrement la partie doit être présente d'ajouter un processus top-down qui irait chercher directement les informations dans les données capteurs. Une étape intermédiaire pourrait être de permettre de changer la qualité des détecteurs en fonction de l'accomplissement de la tâche en cours : ainsi on pourrait diminuer le taux de faux négatifs quitte à augmenter le taux de faux positifs pour être sûr de ne pas manquer une partie lorsque la position est bien établie.

Les différents choix que nous avons réalisés pour notre algorithme ont été présentés. Nous allons expliciter les grandes lignes de l'approche que nous avons développée sachant que le chapitre III décrira de manière plus détaillée l'algorithme.

II.3 Méthode

II.3.1 Principes du système proposé

Le but de notre système va être de mettre à jour un vecteur nommé **vecteur de caractéristiques** à partir de mesures issues d'un ou plusieurs capteurs. En effet, nous transformons le problème de perception en un problème d'estimation. Nous considérons que la reconnaissance d'un objet par exemple, consiste à instancier son vecteur de caractéristiques (position, taille, couleur etc). Cette notion permet d'étendre la perception à d'autres domaines (comme la localisation d'un robot où dans ce cas nous chercherons à instancier sa position). Le système devra solliciter les capteurs selon une approche top-down pour tendre vers la réalisation d'un objectif qui pourra être variable selon l'application : il s'agira par exemple, de déterminer la position d'un véhicule au cours du temps ou encore de reconnaître un objet dans une image.

L'objectif sera ainsi de donner une estimation la plus précise et fiable possible de ce vecteur. La précision se réfère à l'écart qu'il y a entre l'estimation faite pour le vecteur (par exemple une estimation de position) et la réalité. La notion de fiabilité est un peu plus complexe : il est inutile d'avoir une estimation qui affirme être extrêmement précise si cette estimation n'est pas intègre. Ainsi pour une application de localisation, si la position est estimée à 1 cm près mais qu'en réalité, la position réelle se situe 10 mètres plus loin, l'estimation ne peut pas être considérée comme intègre. Il est difficile d'être certain à chaque instant de l'**intégrité** [1] de l'estimation. Un indicateur de confiance va donc être utilisé pour caractériser à quel degré le système espère que l'estimation soit juste. Cette manière de faire présente plusieurs avantages :

— elle peut permettre d'exécuter certaines tâches spécifiques lorsque le système n'est pas sûr de lui : arrêter d'avancer pour un robot mobile, demander une intervention humaine ou simplement chercher à faire d'autres détections.

— elle permet également à l'algorithme de se remettre en question en cherchant la source de ses erreurs.

1. L'intégrité de l'estimation représente le fait que le vecteur de caractéristiques réel est bien compris dans l'ellipse définie à l'aide de l'estimation qui en est faite et de la précision associée à l'estimation.

Enfin, le vecteur de caractéristiques pourra être dynamique : ses caractéristiques évolueront au cours du temps. Il peut s'agir de suivre un véhicule au cours du temps (application de localisation) ou bien un objet dans une image (application de reconnaissance d'objets). Il suffira pour ce faire de connaître un modèle d'évolution du vecteur de caractéristiques. En ce sens, notre système peut être vu comme un estimateur d'état dynamique alimenté par diverses sources d'information sélectionnées de manière active, en vue d'optimiser à la fois la confiance et la précision de l'estimation.

II.3.2 Modélisation

II.3.2.1 Modélisation de l'objet

II.3.2.1.a Récapitulatif des notations pour la modélisation de l'objet

TABLEAU II.2 – Notations pour la modélisation de l'objet

l	indice d'itérations pour la reconnaissance : étape de perception
$\underline{\mathrm{X}}$	vecteur de caractéristiques de l'objet réel de dimension n
n	nombre de composantes du vecteur de caractéristiques de l'objet
$\underline{\mathrm{X}}_{l\|l}$	estimation du vecteur de caractéristiques $\underline{\mathrm{X}}$
$\mathbf{C}_{l\|l}$	matrice de covariance associée à l'estimation $\underline{\mathrm{X}}_{l\|l}$
$\mathrm{P}(\mathbf{E}_{l\|l})$	confiance du système dans l'estimation courante
$\mathbf{E}_{l\|l}$	état de perception à l'étape l : $\mathbf{E}_{l\|l} = \{\underline{\mathrm{X}}_{l\|l}, \mathbf{C}_{l\|l}, \mathrm{P}(\mathbf{E}_{l\|l})\}$
$\mathbf{E}_{l\|l-1}$	état de perception prédit pour l'étape l : $\mathbf{E}_{l\|l-1} = \{\underline{\mathrm{X}}_{l\|l-1}, \mathbf{C}_{l\|l-1}, \mathrm{P}(\mathbf{E}_{l\|l-1})\}$
$\underline{\mathrm{X}}_{l\|l-1}$	estimation prédite du vecteur de caractéristiques $\underline{\mathrm{X}}$
$\mathbf{C}_{l\|l-1}$	matrice de covariance prédite associée à l'estimation $\underline{\mathrm{X}}_{l\|l-1}$
$\mathrm{P}(\mathbf{E}_{l\|l-1})$	confiance prédite du système dans l'estimation prédite
$\mathbf{f}$	fonction d'évolution du vecteur de caractéristiques $\underline{\mathrm{X}}$
$\mathbf{F}$	jacobienne associée à $\mathbf{f}$

II.3.2.1.b Vecteur de caractéristiques de l'objet

Pour modéliser un objet, il est courant de le représenter par un vecteur de caractéristiques, que ce soit dans les modèles basés sur l'apparence [Viola and Jones, 2004, Bourdev and Brandt, 2005, Sankari and Adeli, 2011, Dollár et al., 2012] ou dans les modèles en parties [Leung et al., 1995, Felzenszwalb and McAllester, 2011, Fergus et al., 2003]. On peut ensuite associer à ce vecteur de caractéristiques, soit des classifieurs qui permettront de déterminer si oui ou non l'objet est dans une classe donnée (la plupart du temps avec un score), soit une matrice de covariance permettant de représenter les variations possibles de cet état. Nous nous placerons dans le second cas car cela nous permet d'avoir une représentation explicite de l'objet. Ainsi, un objet sera initialement représenté par un vecteur de caractéristiques $\underline{\mathrm{X}}_{l|l}$ et sa matrice de covariance $\mathbf{C}_{l|l}$ pour l'étape de reconnaissance l. $\underline{\mathrm{X}}_{0|0}$ et $\mathbf{C}_{0|0}$ représenteront alors la connaissance a priori sur le vecteur de caractéristiques et sa matrice de covariance associée.

L'avantage d'avoir un vecteur de caractéristiques noté $\underline{\mathrm{X}}$ est que ces caractéristiques pourront être de nature variable : elles pourront servir à représenter une position, un angle mais aussi une couleur, une forme voire même un son. Chaque composante de ce vecteur de caractéristiques associée à une ou plusieurs autres servira donc à déterminer une propriété de l'objet. Cette représentation a l'avantage d'être générique : il est possible de

représenter un grand nombre d'éléments et de ne pas trop limiter le champ d'application de la perception. Ainsi si nous voulons de la perception plus axée sur la localisation, les composantes de ce vecteur de caractéristiques seront plutôt une position 3D, un angle et une vitesse. Par exemple, ce vecteur pourra être $\underline{X} = \begin{pmatrix} x & y & v_x & v_y & \theta_x & \theta_y \end{pmatrix}^T$ où x et y représenteront la position du véhicule dans le plan, θ_x et θ_y son orientation et v_x, v_y la vitesse selon chaque axe. S'il s'agit de reconnaissance d'objets dans une image, ces composantes représenteront davantage des données de forme, de couleurs et de position pixellique. Par exemple, si l'objet recherché est un carré dans une image, ce vecteur de caractéristiques pourra être $\underline{X} = \begin{pmatrix} c_u & c_v & l \end{pmatrix}^T$ où (c_u, c_v) est le centre du carré dans l'image et l la longueur de ses côtés. Pour modéliser un carré qui ne soit pas nécessairement aligné avec les axes, il est possible de rajouter un angle θ_c permettant de représenter l'orientation du carré dans l'image. Ainsi, en fonction de l'objet recherché, le vecteur de caractéristiques peut être plus ou moins complexe et nécessiter plus ou moins de composantes.

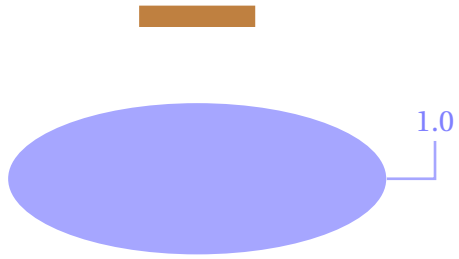
Réussir la tâche de reconnaissance va alors consister à estimer les paramètres optimaux de ce vecteur de caractéristiques $\underline{X}$. Comme pour la perception visuelle humaine, nous proposons une méthode qui déplacera son focus d'attention en fonction de l'état courant de la perception (et donc des valeurs estimées à l'instant courant l du vecteur de caractéristiques $\underline{X}$).

II.3.2.1.c Confiance dans l'estimation

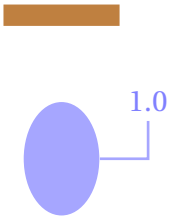
L'objet est donc représenté par un vecteur aléatoire suivant une loi gaussienne qu'il faut probabiliser pour tenir compte des autres hypothèses possibles. La confiance dans l'estimation courante est notée $P(\mathbf{E}_{l|l})$.

La figure II.2 montre différentes situations pour un objet "véhicule". L'objectif est d'estimer sa position dans le plan : $\underline{X} = \begin{pmatrix} X & Y \end{pmatrix}^T$ à partir de l'observation de portes cartographiées. L'estimation de départ est fiable : la confiance dans l'estimation initiale est totale ($P(\mathbf{E}_{0|0}) = 1$) mais peu précise : l'ellipse bleue qui représente l'incertitude sur la position est très large. Nous supposons connaître des **amers** [2] cartographiés dont ici des portes ; en détectant les amers, la position du véhicule pourra être estimée. Dans cet exemple, le véhicule est muni d'un détecteur de porte parfait (la porte est toujours bien détectée si elle est présente et n'est jamais détectée si elle n'y est pas). Ce détecteur nous indique également que le véhicule est en face d'une porte. Dans le premier cas (première ligne), il n'y a qu'une seule porte à détecter donc la détection de la porte permet d'affiner l'estimation sans diminuer la confiance : l'estimation est toujours parfaitement fiable. Dans le second cas (deuxième ligne), il y a deux portes possibles : le véhicule est en face d'une porte, certes mais laquelle ? Une des deux portes est donc choisie. Néanmoins la possibilité de s'être trompé est prise en compte. L'estimation est affinée mais la confiance a diminué : pour la pose choisie (ellipse bleue toujours), elle n'est plus que de la moitié ($P(\mathbf{E}_{1|1}) = 0.5$), l'autre moitié étant prise par l'autre position possible (ellipse rose). Or nous ne faisons pas de multi hypothèses à cause notamment de la combinatoire liée aux choix successifs qui peut engendrer des temps de calculs prohibitifs. Une seule hypothèse est donc traitée à la fois. L'hypothèse en cours est alors probabilisée pour tenir compte des erreurs potentielles liées entre autres à ces problèmes d'ambiguïté.

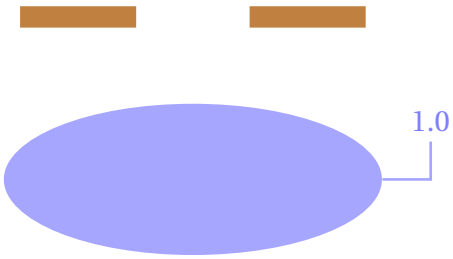
2. un amer est un point de repère fixe et identifiable sans ambiguïté utilisé pour la navigation maritime. Le terme amer est également utilisé en robotique mobile pour désigner les points de repère utilisés par un robot pour se repérer dans son environnement et calculer sa trajectoire.



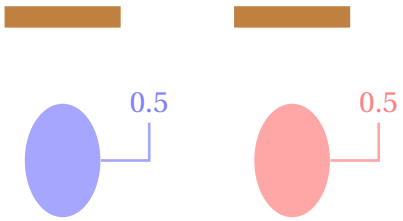
(a) Avant détection de la porte : estimation peu précise et sans ambiguïté en présence d'une seule porte.



(b) Après détection de la porte : estimation plus précise et sans ambiguïté



(c) Avant détection de la porte : estimation peu précise et sans ambiguïté en présence de deux portes.



(d) Après détection de la porte : estimation plus précise mais ambiguë

FIGURE II.2 – Différentes situations d'estimation pour un véhicule à l'aide de la détection de portes. La première ligne représente une situation où il y a une seule et unique porte à détecter. Pour la seconde ligne, deux portes sont présentes. Dans les deux cas, au départ (colonne de gauche), l'estimation (ellipse bleue) est très peu précise mais fiable ($P(\mathbf{E}_{0|0}) = 1$). Le numéro à proximité de chaque ellipse indique la confiance dans l'estimation. La deuxième colonne représente la nouvelle estimation après la détection d'une porte (représentée par les lignes marrons). Dans le premier cas, l'estimation demeure fiable car il n'y a qu'une seule porte. Dans le deuxième cas, la confiance diminue : il y a une chance sur deux d'avoir choisi la bonne porte.

II.3.2.1.d État de l'objet

L'état courant de perception $\mathbf{E}_{l|l}$ est défini par $\mathbf{E}_{l|l} = \{\underline{X}_{l|l}, \mathbf{C}_{l|l}, P(\mathbf{E}_{l|l})\}$ avec :

— $\underline{X}_{l|l}$ est l'estimation de $\underline{X}$ à cette étape l de perception.
— $\mathbf{C}_{l|l}$ est la matrice de covariance associée.
— $P(\mathbf{E}_{l|l})$ est la confiance à cette étape.

$\underline{X}_{l|l}$ est l'estimation de la perception : c'est la représentation courante de l'objet en fonction des détections précédentes. À la toute première étape, avant toute détection, il s'agit donc d'une représentation de l'objet *moyen*. $\mathbf{C}_{l|l}$ permet quant à elle de représenter la précision de l'estimation courante. $P(\mathbf{E}_{l|l})$, la confiance de l'estimation courante, caractérise l'intégrité de l'estimation $\underline{X}_{l|l}$, c'est à dire la probabilité que les paramètres réels du vecteur de caractéristiques $\underline{X}$ suivent une loi gaussienne définie par $\underline{X}_{l|l}$ et $\mathbf{C}_{l|l}$. En d'autres mots, c'est la probabilité que la **vraie** position $\underline{X}$ soit dans l'espace défini par $\underline{X}_{l|l}$ et $\mathbf{C}_{l|l}$.

II.3.2.1.e Objectifs de reconnaissance

Le petit exemple du véhicule détectant des portes permet d'appréhender quel va alors être l'objectif de notre système de perception : il s'agit de déterminer l'état optimal $\mathbf{E}_{opt|opt}$

où opt est la valeur de l ayant conduit à la meilleure estimation. $\mathbf{E}_{opt|opt}$ est donc l'état qui maximise à la fois la précision sur l'estimation de $\underline{X}$ et la confiance $P(\mathbf{E}_{opt|opt})$. Donc au contraire des méthodes d'estimation classique (de type Kalman par exemple) où seul un critère de précision entre en compte pour estimer l'état courant d'un système, ici, nous intégrerons aussi **la notion de confiance**. Ce sont ces deux paramètres qui composeront l'objectif à atteindre. C'est donc un objectif double qui va comporter à la fois une partie sur la précision de l'estimation, appelée *objectif de précision* et qui sera noté $\mathbf{C}_{Obj}$ et un autre volet qui cherchera à maximiser la confiance et qui sera nommé *objectif de fiabilité ou d'intégrité* et noté P_{Obj}. Nous reviendrons sur ces objectifs dans le § III.3.2.

Ce double aspect permet de bien représenter différentes situations de la perception réelle humaine. Par exemple, il est possible d'avoir seulement aperçu du coin de l'œil un "objet" et soit ne pas être capable de l'identifier : "j'ai vu un éclat rouge, c'est peut-être un ballon" ; soit à l'inverse avoir pu l'identifier mais être incapable de donner des caractéristiques de l'objet : "j'ai vu l'ombre d'un chat" (c'est un chat, c'est sûr, par contre je ne peux pas apporter beaucoup de précisions dans l'espace "objet" chat : est-il grand, gros, petit, noir, blanc ou autre ?). Pour des applications de type localisation, cela semble aussi adapté. Dans l'exemple précédent des deux portes, la précision a pu être affinée tout en diminuant la confiance. À l'inverse, au départ, la confiance est grande mais la précision est très faible. Les deux sont complémentaires : dire "je connais ma position à 1 mm près mais il y a 1% de chance que ce soit exact" n'est pas très encourageant ; à l'inverse dire "je suis sur Terre mais j'en suis absolument certain" n'a pas grand intérêt non plus.

II.3.2.1.f Prédiction de l'état de l'objet

Nous aurons besoin de prédire l'état courant du système en fonction de l'état précédent. Nous noterons $\mathbf{E}_{l|l-1}$, l'état prédit pour l'étape l de reconnaissance. Pour déterminer cet état prédit, nous admettons disposer d'une fonction d'évolution $\mathbf{f}$ et de sa jacobienne associée $\mathbf{F}$ pour chaque application. Par exemple, pour un véhicule se déplaçant le modèle d'évolution pourra être calculé grâce à l'odométrie du véhicule. Pour une application de reconnaissance d'objets dans une image, la fonction d'évolution sera l'identité : l'objet reste à l'identique tant que la même image est traitée.

Pour le vecteur de caractéristiques $\underline{X}_{l|l-1}$ et sa covariance associée $\mathbf{C}_{l|l-1}$, ils seront obtenus à l'aide des équations de prédiction présentées en annexe B. Pour la confiance associée à l'état prédit, $P(\mathbf{E}_{l|l-1})$, elle sera supposée identique à celle de l'état précédent :

$$P(\mathbf{E}_{l|l-1}) = P(\mathbf{E}_{l-1|l-1})$$

Nous venons donc de présenter la modélisation de l'objet dans sa globalité. Maintenant, comme annoncé dans le § II.2, nous avons choisi de découper l'objet en différentes parties. Nous allons donc expliquer dans le paragraphe suivant comment nous modéliserons ces parties.

II.3.2.2 Modélisation des parties

II.3.2.2.a Récapitulatif des notations pour la modélisation des parties

TABLEAU II.3 – Notations pour la modélisation des parties

M	nombre de parties de l'objet
i	indice pour la $i$ème partie de l'objet, $i \in [1, \mathrm{M}]$.
$\underline{\mathrm{Y}}^i$	vecteur de caractéristiques réel associé à la partie i.
$\underline{\mathrm{Y}}^i_{l\|l}$	estimation du vecteur de caractéristiques associé à la partie i.
$\Sigma^i_{l\|l}$	matrice de covariance associée.
$\mathrm{P}(\mathrm{A}^i_{l\|l})$	probabilité d'existence de la partie.
$\eta^i_{l\|l}$	état de la partie i à l'étape de perception l : $\eta^i_{l\|l} = \{\underline{\mathrm{Y}}^i_{l\|l}, \Sigma^i_{l\|l}, \mathrm{P}(\mathrm{A}^i_{l\|l})\}$
$\eta^i_{l\|l-1}$	état prédit de la partie i à l'étape de perception l : $\eta^i_{l\|l-1} = \{\underline{\mathrm{Y}}^i_{l\|l-1}, \Sigma^i_{l\|l-1}, \mathrm{P}(\mathrm{A}^i_{l\|l-1})\}$
$\underline{\mathrm{Y}}^i_{l\|l-1}$	vecteur de caractéristiques prédit associé à la partie i.
$\Sigma^i_{l\|l-1}$	matrice de covariance prédite associée.
$\mathrm{P}(\mathrm{A}^i_{l\|l-1})$	probabilité d'existence prédite de la partie.
$\mathbf{h}_i$	fonction permettant de relier le vecteur de caractéristiques de la partie $\underline{\mathrm{Y}}$ au vecteur de caractéristiques de l'objet $\underline{\mathrm{X}}$: $\underline{\mathrm{Y}}^i = \mathbf{h}_i(\underline{\mathrm{X}})$
$\mathbf{H}_i$	jacobienne associée à $\mathbf{h}_i$

II.3.2.2.b Découpage en parties

L'avantage principal de décomposer un objet en parties est de pouvoir représenter des objets très déformables ou des classes d'objets très variables. En effet, si l'apparence de l'objet dans sa globalité peut être très changeante, en revanche, il est très souvent possible de décomposer l'objet en parties peu déformables. Par contre, les liens entre ces parties sont susceptibles de changer (comme des angles ou autre). Ainsi si un piéton vu de face tend les bras ou les a le long du corps, cela pourra modifier de manière importante son apparence globale alors que ses bras ou sa tête seront eux peu modifiés.

Un autre avantage de ce découpage en parties peut être de permettre la focalisation qu'elle soit spatiale ou de caractéristiques. Si un œil gauche vert a été trouvé, il est probable que l'œil droit soit vert aussi, de même s'il fait une certaine taille dans l'image, l'autre œil devrait avoir une taille similaire. Pour la focalisation spatiale, si les deux yeux ont été trouvés, les positions envisageables pour la bouche sont assez restreintes, pour avoir un visage cohérent. Une telle possibilité n'existe pas pour les approches bottom-up qui remonteraient toutes les correspondances avec un œil standard.

Deux points importants sont à noter pour ce découpage en parties. D'une part, il faudra qu'une grande majorité des parties définies soient détectables. Pour la suite, seules les parties détectables nous intéresseront. Il devra exister au moins un détecteur capable de trouver la partie dans une certaine modalité capteur. D'autre part, la connaissance des parties devra pouvoir nous renseigner sur l'état de l'objet : il faudra donc disposer d'un lien entre la représentation de la partie et le vecteur de caractéristiques de l'objet (il s'agira de la fonction $\mathbf{h}_i$).

La détermination des parties (et des éléments nécessaires à leur modélisation) est donc laissée au choix de l'utilisateur. Une telle modélisation peut être coûteuse à mettre en

place et ne garantit aucunement la pertinence de la partie choisie. Ainsi, pour détecter un piéton, il ne parait pas très judicieux de définir une partie *orteil*. En contrepartie, il n'y a pas réellement de limitation sur le type de parties qu'il est possible de prendre en compte, ni sur leur nombre ou leur complexité. Il peut donc y avoir des parties très simples (un "cercle", un "point", une "ligne") et d'autres beaucoup plus complexes.

L'objet sera composé de M parties. Il peut s'agir de parties liées à la forme de l'objet : un bras, une tête ou une jambe pour une personne, ou à l'apparence : des yeux bleus, verts ou gris ou à n'importe quel ensemble de caractéristiques qui assemblées forment un tout identifiable (nous rappelons qu'il existe au moins un détecteur qui pour une modalité capteur donné est capable de détecter la partie).

II.3.2.2.c État d'une partie

Chaque partie i sera elle aussi représentée par un état

$$\eta^i_{l|l} = \{\underline{Y}^i_{l|l}, \Sigma^i_{l|l}, P(A^i_{l|l})\}$$

où

- $\underline{Y}^i_{l|l}$ est le vecteur de caractéristiques associé à la partie i pour un niveau de reconnaissance l.
- $\Sigma^i_{l|l}$ est la matrice de covariance associée.
- $P(A^i_{l|l})$ est la probabilité d'existence de la partie.

Comme pour l'objet, la partie est modélisée par une gaussienne. Semblablement, il faudra estimer l'état de la partie $\eta^i_{l|l}$.

$P(A^i_{l|l})$ permet de représenter des parties optionnelles comme des lunettes pour des piétons. Si la partie est toujours présente (comme les roues pour des voitures en état normal), alors $P(A^i_{l|l}) = 1$ et ce pour tout l : l'existence de la partie est attestée. Mais de manière générale, cette probabilité d'existence pourra évoluer avec le temps : par exemple, si à une étape initiale ($l = 0$), le pourcentage de personnes qui portent des lunettes est estimé à 20% alors $P(A^{lunettes}_{0|0}) = 0.2$ mais si quelques étapes plus tard, plusieurs détecteurs de lunettes ont été utilisés et qu'aucun n'a détecté de lunettes pour la personne en cours de détection, la probabilité tendra vers 0. À l'inverse, si la plupart ont trouvé des lunettes, la probabilité aura augmenté et tendra vers 1.

Nous faisons l'hypothèse d'avoir pour chaque partie i, une fonction $\mathbf{h}_i$ qui relie le vecteur de caractéristiques $\underline{Y}^i$ de la partie au vecteur de caractéristiques de l'objet $\underline{X}$:

$$\underline{Y}^i = \mathbf{h}_i(\underline{X}) \tag{II.1}$$

L'estimation du vecteur de caractéristiques devrait donc pouvoir s'écrire semblablement à un bruit de mesure b_{mes} près (lié au capteur et au détecteur) :

$$\underline{Y}^i_{l|l} = \mathbf{h}_i(\underline{X}_{l|l}) + b_{mes} \tag{II.2}$$

Cette fonction est supposée connue à l'avance. Différentes parties pourront avoir des vecteurs de caractéristiques $\underline{Y}^i_{l|l}$ de taille différente (une partie peut donc avoir plus de paramètres qu'une autre). Les parties d'un même objet sont donc reliées entre elles uniquement par le vecteur de caractéristiques complet de l'objet $\underline{X}_{l|l}$. Rajouter (ou enlever)

une partie n'aura donc aucune influence sur les autres parties et ne modifiera aucunement les autres fonctions $\mathbf{h}_i$.

La matrice de covariance $\Sigma^i_{l|l}$ est donnée par l'approximation suivante :

$$\Sigma^i_{l|l} = \mathbf{H}_i \mathbf{C}_{l|l} \mathbf{H}_i^{\mathrm{T}} + \mathrm{Q}_{mes} \tag{II.3}$$

où $\mathbf{H}_i$ est la matrice jacobienne de la fonction $\mathbf{h}_i$ et Q_{mes} est la covariance du bruit de mesure b_{mes}. Une autre approximation aurait pu être choisie, par exemple en s'inspirant du filtre Kalman sans parfum [Wan and Van Der Merwe, 2000]. $\mathbf{H}_i$ est, elle aussi, supposée connue ou du moins, il est admis qu'une méthode de calcul pour l'obtenir à partir de $\mathbf{h}_i$ est disponible et qu'elle est donc calculable. $\Sigma^i_{l|l}$ représente la dispersion des paramètres de la partie connaissant l'estimation $\underline{\mathrm{X}}_{l|l}$ des paramètres de l'objet. Pour la probabilité d'existence de la partie $\mathrm{P}(\mathrm{A}^i_{l|l})$, nous admettons semblablement que son évolution dépendra de la confiance dans l'état $\mathbf{E}_{l|l}$ mais les mécanismes pour l'estimer seront présentés dans le § III.5.3. L'état $\eta^i_{l|l}$ est donc dépendant de l'estimation courante de l'état de l'objet.

II.3.2.2.d Prédiction de l'état d'une partie

Comme pour l'état de l'objet, il nous sera utile de prédire l'état d'une partie pour l'étape de reconnaissance l connaissant son état à l'étape $l-1$. Nous noterons $\eta^i_{l|l-1}$ cet état prédit.

Pour la prédiction du vecteur de caractéristique $\underline{\mathrm{Y}}^i_{l|l-1}$ et la matrice de covariance associée $\Sigma^i_{l|l-1}$, nous utiliserons l'état prédit de l'objet $\mathbf{E}_{l|l-1}$ (voir le § II.3.2.1.f) et les approximations précédentes :

$$\begin{aligned} \underline{\mathrm{Y}}^i_{l|l-1} &= \mathbf{h}_i(\underline{\mathrm{X}}_{l|l-1}) + b_{mes} \\ \Sigma^i_{l|l-1} &= \mathbf{H}_i \mathbf{C}_{l|l-1} \mathbf{H}_i^{\mathrm{T}} + \mathrm{Q}_{mes} \end{aligned}$$

Pour la probabilité d'existence de la partie, il n'y a, a priori, aucune raison pour qu'une partie apparaisse ou disparaisse subitement donc la valeur de l'étape précédente sera conservée :

$$\mathrm{P}(\mathrm{A}^i_{l|l-1}) = \mathrm{P}(\mathrm{A}^i_{l-1|l-1})$$

Maintenant que nous avons présenté la modélisation générale de l'objet et des parties qui le composent, nous allons étudier l'algorithme proposé. Dans un premier temps, nous présenterons le schéma global de l'approche proposée puis nous expliquerons le principe général régissant chacune des étapes.

II.3.3 Principe général de notre algorithme

La figure II.3 montre les différentes étapes de notre algorithme. Nous avons tout d'abord une phase d'initialisation qui est réalisée une seule fois au début. Puis nous cherchons parmi l'ensemble des parties, des détecteurs et des capteurs (qui forment donc des triplets comme nous le verrons plus en détails dans le § II.3.3.2), lequel parait être le meilleur. Une fois ce choix réalisé, une détection est tentée. L'état de l'objet est ensuite mis à jour en fonction de cette détection. À l'issue de cette mise à jour, soit l'objet est reconnu et donc l'algorithme se termine et le résultat est sauvegardé. Soit la reconnaissance n'est pas encore achevée et les étapes de sélection, détection et mise à jour seront de nouveau exécutées. Ces différentes étapes vont dans un premier temps être décrites sommairement afin de saisir le principe général qui les anime puis nous y reviendrons plus en détails dans le chapitre III.

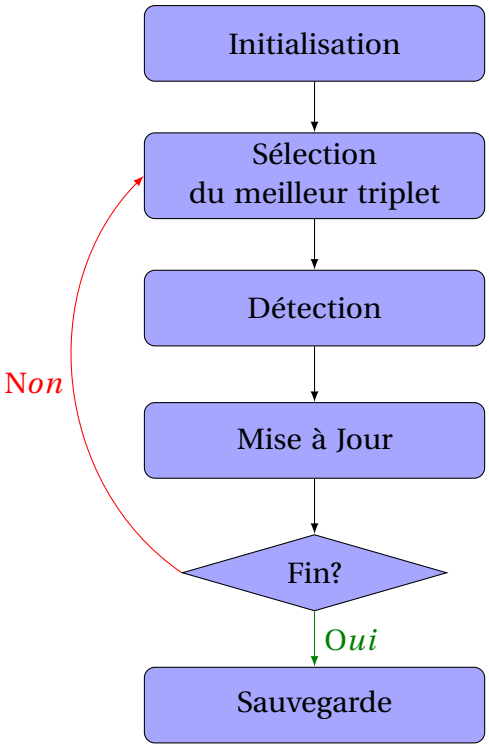


FIGURE II.3 – Schéma très simplifié du fonctionnement de notre algorithme. La notion de triplets sera vue dans le § II.3.3.2.

II.3.3.1 Initialisation

L'objectif de cette étape est de définir toutes les valeurs a priori nécessaires au bon fonctionnement de l'algorithme. Notamment, l'état initial $\mathbf{E}_{0|0} = \{\underline{X}_{0|0}, \mathbf{C}_{0|0}, P(\mathbf{E}_{0|0})\}$ est connu : la valeur moyenne initiale du vecteur de caractéristiques, la matrice de covariance initiale (et donc les variations admises pour les différents paramètres) ainsi que la confiance initiale sont données. Au départ, si peu d'informations sont disponibles, la matrice de covariance peut potentiellement avoir des valeurs conséquentes représentant ainsi une grande variabilité pour chacun des paramètre de $\underline{X}_{0|0}$. De même, en l'absence totale de connaissance, $P(\mathbf{E}_{0|0})$ peut être initialisée à la valeur de 0.5 ce qui indique donc qu'il n'y a pas de connaissance a priori sur la présence ou l'absence de l'objet dans la scène.

C'est également lors de cette étape initiale que pour chaque partie i, les probabilités $P(A^i_{0|0})$ sont initialisées. Néanmoins, ces valeurs pourront varier lors des itérations suivantes de l'algorithme. Par exemple, en admettant que la probabilité d'avoir des lunettes pour une personne soit de 20%, la probabilité associée à la partie *lunettes* commencera donc à la valeur de 0.2 et pourra évoluer soit vers 0 indiquant que la personne perçue n'a pas de lunettes ou à l'inverse vers 1 si elle en possède.

En résumé, c'est dans cette étape que toutes les connaissances a priori sur l'objet sont mobilisées et données à notre algorithme via différentes valeurs comme les probabilités mentionnées précédemment.

II.3.3.2 Notion de triplet perceptif

Tout d'abord définissons la notion de triplet perceptif qui est issue de [Tessier et al., 2009]. Notre ambition est de définir un système de perception générique et potentiel-

lement multi-capteurs. Un triplet perceptif T_j est l'ensemble constitué d'une partie P_j, d'un capteur Ca_j et d'un détecteur D_j soit $T_j = \{P_j, Ca_j, D_j\}$. En effet, une partie peut être perçue par différents capteurs : ainsi une partie *mur* peut être perçue via une caméra (renvoyant une image contenant le mur visible) ou à l'aide d'un télémètre laser (qui renverra des points appartenant potentiellement au mur). Il sera donc possible d'avoir un triplet $T_1 = \{P_1, Ca_1, D_1\}$ et un triplet $T_2 = \{P_2, Ca_2, D_2\}$ avec $P_1 = P_2$, $Ca_1 \neq Ca_2$ (et donc $D_1 \neq D_2$ mais l'aspect détecteur sera détaillé ensuite) : les deux triplets sont liés à la même partie mais avec des capteurs différents. Tous les capteurs ne seront pas adéquats pour toutes les parties. Par exemple, un télémètre laser sera sans intérêt pour détecter une partie correspondant uniquement à de la couleur.

Pour chaque partie, avec chaque capteur compatible, il faut un détecteur capable d'identifier la partie. Ainsi dans l'exemple du mur précédent, avec le capteur télémètre laser, il nous faudra un détecteur de lignes. Pour un même couple partie-capteur, plusieurs détecteurs différents peuvent coexister. En effet, toujours avec l'exemple du mur, plusieurs détecteurs de lignes existent et pourront donc potentiellement être utilisés par notre algorithme. Il sera donc possible d'avoir un triplet $T_{1'} = \{P_{1'}, Ca_{1'}, D_{1'}\}$ et un triplet $T_{2'} = \{P_{2'}, Ca_{2'}, D_{2'}\}$ avec $P_{1'} = P_{2'}$, $Ca_{1'} = Ca_{2'}$ et $D_{1'} \neq D_{2'}$: seul le détecteur diffère entre les deux triplets. Tous les détecteurs n'offrent pas le même temps de réponse, ni la même qualité que ce soit en terme de fiabilité ou de précision. Avec notre approche, nous souhaitons solliciter le meilleur détecteur selon la situation courante.

La liste de tous les triplets possibles est supposée connue (c'est à dire que pour chaque partie il est possible d'associer un ou plusieurs capteurs et pour chaque couple ainsi formé de donner au moins un détecteur compatible).

II.3.3.3 Sélection du meilleur triplet perceptif

Rappelons que l'algorithme est séquentiel et mono-hypothèse : une partie est sélectionnée puis détectée, l'état estimé de l'objet est mis à jour et ainsi de suite. Mais, nous voulons choisir efficacement non seulement la partie à détecter mais aussi comment la détecter (dans quel capteur et avec quel détecteur). Donc la question est de savoir comment faire ce choix de la manière la plus efficace possible en fonction de la tâche à accomplir, de l'objectif à atteindre et et de l'état courant de l'estimation de l'objet.

Le problème revient donc à rechercher quel triplet perceptif doit être sollicité à chaque instant. En effet, compte tenu de la position de l'objet, certaines parties peuvent ne pas être pertinentes , par exemple car elles ne sont pas visibles avec les capteurs disponibles. En fonction de l'état de reconnaissance, il pourra être plus pertinent de sélectionner un triplet plutôt qu'un autre. Le choix est donc dynamique. Par exemple, admettons que nous souhaitions déterminer la position d'un véhicule à l'aide d'une carte. Au départ, le véhicule est positionné de manière certaine dans une forêt avec une grande incertitude de position (disons plus d'1 km). Il parait peu pertinent de commencer à se localiser à l'aide d'un détecteur d'arbres : même si celui-ci trouve un arbre, il sera difficile de dire de quel arbre de la forêt il s'agit. À l'inverse dans cette situation, le récepteur GPS, capable de donner une estimation à 1 m près, sera très pertinent car il précisera grandement la position. Si au contraire, l'estimation est déjà précise à 50 cm près, le récepteur GPS ne sera pas pertinent puisqu'il ne nous apportera aucune information supplémentaire, le détecteur d'arbre deviendra quant à lui plus intéressant.

L'objectif de cette étape est donc de définir un critère générique pouvant s'appliquer à tous les triplets. Le meilleur triplet sera choisi en fonction des circonstances actuelles et des objectifs à atteindre. Le choix du meilleur triplet peut être également vu comme un choix qui nous permettra de progresser au mieux vers les objectifs fixés. Certains triplets sont inintéressants. Certains n'améliorent pas l'estimation de l'état de l'objet, d'autres ont des détecteurs qui ne sont pas assez fiables ou trop lents. Pour chaque triplet, un critère représentant son apport informationnel probable sera calculé. Le triplet ayant le critère le plus élevé sera alors choisi.

II.3.3.4 Détection

II.3.3.4.a Focalisation Spatiale

Une fois le triplet sélectionné, le détecteur du triplet va être exécuté dans une zone de l'image capteur du capteur sélectionné. En effet, plutôt que de traiter toute l'image capteur, seule la zone *intéressante* sera étudiée. Nous utilisons donc ici la notion de *focalisation spatiale* dans l'espace capteur. Celle-ci amènera à la fois une meilleure précision mais aussi diminuera la probabilité de détection d'outliers et décrémentera les temps de calcul (puisque le détecteur sera appelé dans une zone plus petite).

II.3.3.4.b Focalisation de caractéristiques

Dans la ROI déterminée précédemment via la *focalisation spatiale*, le détecteur est appelé. Il va rechercher des primitives compatibles avec la partie de l'objet recherché. Nous utilisons donc ici la notion de *focalisation de caractéristiques*. Là aussi, cela permettra de diminuer la probabilité de détecter des outliers.

Le système sera donc en mesure de focaliser de manière optimale que ce soit dans l'espace capteur (*focalisation spatiale*) ou dans l'espace de caractéristiques (*focalisation de caractéristiques*) à chaque étape du processus de reconnaissance.

II.3.3.4.c Échec ou Réussite de la Détection

Si aucune primitive n'a été renvoyée par le détecteur, il est aisé de voir que la détection a échoué. À l'inverse, s'il y a au moins une primitive compatible avec la partie cherchée alors la détection a réussi. Lorsque le détecteur renvoie une liste de primitives, il faut donc se poser la question de la compatibilité avec la partie cherchée, en particulier dans le cas des détecteurs non paramétrables. Pour savoir si la primitive est compatible, une distance de Mahalanobis sera calculée entre le vecteur de caractéristiques de la partie et le vecteur de caractéristiques de la primitive. Si cette distance est inférieure à un seuil S alors la primitive sera compatible. Il peut donc arriver que bien que le détecteur ait renvoyé des primitives, finalement aucune ne convienne car aucune n'est compatible avec la partie cherchée ; la détection a finalement échoué. Un signal *échec* ou *réussite* sera donc renvoyé et sera utilisé par la suite notamment dans l'étape de mise à jour.

II.3.3.4.d Sélection de la primitive retenue

Il est possible que le détecteur renvoie plusieurs primitives compatibles avec la partie recherchée. Dans ce cas-là, une seule primitive sera sélectionnée : la primitive la plus proche (au sens toujours de la distance de Mahalanobis) du vecteur de caractéristiques prédit pour la partie $\underline{Y}^i_{l|l-1}$ (voir le § II.3.2.2.d) sera choisie.

II.3.3.5 Mise à Jour

Une fois le processus de détection terminé, l'état va être mis à jour. L'état $\mathbf{E}_{l|l}$ est en fait l'état mis à jour après le processus de détection. L'état prédit $\mathbf{E}_{l|l-1}$ peut donc être vu comme l'état à l'étape l avant la mise à jour.

La mise à jour est découpée en deux étapes distinctes : une mise à jour *de la confiance* $P(\mathbf{E}_{l|l})$ et une mise à jour de *position* qui donc sera chargée de calculer le nouveau vecteur de caractéristiques $\underline{X}_{l|l}$ et la covariance associée $\mathbf{C}_{l|l}$.

II.3.3.5.a Mise à jour de la confiance

Ici, l'objectif est de calculer $P(\mathbf{E}_{l|l})$ en fonction du résultat de détection. La valeur de $P(\mathbf{E}_{l|l})$ dépendra de :

- la confiance dans l'état avant détection $P(\mathbf{E}_{l|l-1})$: selon la confiance précédente, la faute d'une non détection pourra au choix être plutôt attribuée à une erreur du détecteur (l'estimation était fiable) ou à une erreur sur l'estimation.
- la qualité du détecteur :
  - la probabilité que le détecteur ne *voit* pas la partie alors qu'elle est bel et bien présente (taux de faux négatifs). Si ce taux est élevé, ne pas voir la partie sera bien moins inquiétant que si ce taux est très faible.
  - la probabilité que le détecteur *voit* la partie alors qu'elle n'y est pas (taux de faux positifs). Si ce taux est faible, trouver une primitive compatible avec la partie sera plus rassurant que si ce taux est élevé.
- la qualité de l'information :
  - est-ce qu'il existe ailleurs dans *le monde* des primitives correspondant à la partie cherchée ? Est-il possible de trouver une primitive compatible alors que l'estimation n'est pas intègre ? Si oui, quelle est la probabilité associée à cet événement ? Plus la partie est unique dans le monde, plus le fait de la trouver renforcera la confiance dans l'estimation.
  - est-ce qu'il y a dans le champ de perception du détecteurs, d'autres primitives compatibles, et si oui combien, ce qui correspond à quelle est la probabilité de choisir la bonne parmi toutes celles qui sont compatibles ?

Afin de calculer la nouvelle confiance, un outil mathématique permettant de gérer des probabilités est indispensable. Nous le présenterons dans la section III.4. Nous reviendrons plus en détails sur les différents éléments dans le § III.6.1. Ici, nous admettons que nous serons capables de calculer une nouvelle confiance en fonction des différents éléments proposés ci-dessus.

II.3.3.5.b Mise à jour de l'estimation

Tout d'abord, la mise à jour de *l'estimation*, c'est à dire du vecteur $\underline{X}_{l|l-1}$, ne pourra avoir réellement lieu que si la détection a réussi. En effet, c'est le seul cas dans lequel la partie recherchée a été instanciée. Donc une primitive compatible a été trouvée. Dès lors, pour la partie concernée, au lieu d'avoir uniquement une hypothèse sur la valeur de ses caractéristiques, une estimation de ses caractéristiques réelles est disponible. Par exemple, pour un carré positionné par défaut au centre de l'image, une fois un de ces côtés détecté, il est possible d'en déduire la position estimée du centre du carré.

Pour cette mise à jour, une équation dynamique bayésienne (un filtre de Kalman dans notre cas) nous permettra de déduire la nouvelle position estimée de l'objet ($\underline{X}_{l|l}$ et $\mathbf{C}_{l|l}$) en fonction de l'état prédit ($\underline{X}_{l|l-1}$ et $\mathbf{C}_{l|l-1}$) et de la mesure $\underline{Y}^{i}_{l|l}$ (ici la primitive sélectionnée pour la partie cherchée).

II.3.3.5.c Mise à jour de retour

Il est possible d'avoir fait une mauvaise association lors d'une étape précédente r : il existait plusieurs primitives compatibles et celle sélectionnée s'avère ne pas être la bonne (elle ne correspond pas à la partie cherchée). Par exemple, plusieurs segments verticaux dans une image ont pu être trouvés lors de la recherche d'une partie *bordGauche* pour un carré aligné avec les axes. Le segment n°1 est sélectionné en premier mais il n'appartenait pas au carré. L'idée est alors de revenir sur "nos pas" et de choisir cette fois-ci le segment n°2 (et ainsi de suite jusqu'à la sélection du bon segment).

Pour cette mise à jour de retour, il faut en premier lieu déterminer de manière automatique à quel moment il est nécessaire de la faire. Cela dépendra de la confiance actuelle et des résultats des détections : tant que les détections réussissent, il n'y a a priori aucune raison de revenir en arrière. Semblablement, si la confiance en l'estimation est bonne, il ne semble pas judicieux de tout remettre en cause.

Quand cette mise à jour de retour s'avère nécessaire, il faut ensuite faire le retour en arrière en revenant à une itération (r) où il y avait eu plusieurs choix possibles (plusieurs primitives compatibles) pour une partie donnée. L'objectif est d'offrir l'occasion à l'algorithme de tester une autre primitive afin d'espérer sélectionner la "bonne".

Avec la *focalisation spatiale* et la *focalisation de caractéristiques* au fur et à mesure des itérations, il devrait y avoir de moins en moins de candidats possibles pour une partie donnée, en conséquence les retours en arrière auront surtout lieu au niveau des premières itérations. En effet, c'est au début du processus de reconnaissance que les focalisations ont le moins d'influence puisqu'il y a encore trop peu de données pour avoir resserrer l'estimation. En conséquence, il y a plus de risques de détecter plusieurs primitives compatibles pour une même partie et donc de faire un mauvais choix nécessitant un retour.

Les différents processus de mise à jour qui interviennent dans notre algorithme ont été succinctement présentés. Une fois cette mise à jour terminée, il faut interroger l'algorithme pour savoir s'il a terminé.

II.3.3.6 Critère de fin

Dans le § II.3.2.1.e , deux objectifs ont été définis : un objectif de précision noté $\mathbf{C}_{Obj}$ et un objectif de confiance noté P_{Obj}. Si le vecteur de caractéristiques de l'objet comporte n caractéristiques, $\mathbf{C}_{Obj}$ sera donc une matrice de taille $n \times n$ indiquant pour chaque caractéristiques la variance autorisée. P_{Obj} correspondra à la confiance minimale souhaitée dans l'estimation. L'algorithme se terminera si ces deux objectifs sont atteints. C'est le cas normal de terminaison de l'algorithme. Il existe d'autres cas particuliers qui impliqueront une sortie de l'algorithme : ils seront décrits dans le § III.7. Une fois ce critère de fin atteint, l'algorithme se termine.

II.4 Bilan

Dans ce chapitre, nous avons défini les objectifs de notre algorithme : nous souhaitons développer un système de perception générique pouvant s'adapter à différentes applications. Il doit pouvoir être multi-capteurs et être modulaire, notamment permettre l'ajout de nouveaux détecteurs en fonction des besoins. Nous souhaitons qu'il soit capable d'utiliser les connaissances a priori dont il peut disposer pour la tâche à accomplir. De plus, en fonction de l'état courant du système par rapport à l'objectif visé, il doit être capable de sélectionner les données à la fois en cherchant dans des zones intéressantes (ROI) mais aussi en acceptant uniquement les données pertinentes. Enfin, en cas d'erreur, il doit être capable de se remettre en cause et de revenir en arrière si nécessaire.

Compte tenu de ces objectifs, les méthodes top-down nous ont paru être plus adaptées en particulier pour la gestion des connaissances a priori. Nous avons sélectionné parmi ces méthodes top-down les méthodes impliquant un modèle en parties afin de permettre de focaliser comme nous le souhaitions. Nous avons préféré un découpage en parties manuel qui s'il est plus complexe à mettre en place, est par contre plus modulaire et permet plus aisément la gestion des occultations et l'ajout de nouvelles parties. Nous avons aussi choisi de représenter notre problème comme un problème d'estimation, ce qui donnera son caractère générique à notre algorithme.

Une fois ces choix effectués, nous avons présenté la modélisation proposée pour l'objet à l'aide d'un état $\mathbf{E}_{l|l}$ comportant à la fois des données sur l'estimation ($\underline{\mathrm{X}}_{l|l}$ et $\mathbf{C}_{l|l}$) et sur la confiance que le système a dans cette estimation ($\mathrm{P}(\mathbf{E}_{l|l})$). Nous avons également modélisé les différentes parties qui composent cet objet à l'aide, là aussi, d'un état $\eta^i_{l|l}$. Des fonctions $\mathbf{h}_i$ sont utilisées afin de représenter les liens entre le vecteur de caractéristiques de l'objet et les vecteurs de caractéristiques de chaque partie i.

Ensuite, le principe général de notre algorithme a été présenté. L'algorithme sélectionne, à chaque étape, le meilleur triplet perceptif. Un détecteur est ensuite appelé dans la région d'intérêt définie à partir des informations disponibles à cette étape. Les primitives compatibles avec la partie cherchée sont extraites. S'il y a plusieurs primitives compatibles, une seule est sélectionnée. En fonction du résultat de la détection (échec ou réussite) et de l'éventuelle primitive trouvée, l'état global de l'objet est mis à jour. Après la mise à jour, si les objectifs de précision et de confiance sont remplis, l'algorithme se termine sinon un nouveau triplet sera sélectionné. Les grands principes régissant chacune de ces étapes ont été expliqués.

Dans le chapitre III, nous allons donc présenter de manière détaillée ces différentes étapes.

Chapitre III

Algorithme de perception développé

« Il est dans la probabilité que mille choses arrivent qui sont contraires à la probabilité. »

Henri Louis Mencken

Sommaire

III.1 Rappel

Nous avons déjà décrit sommairement notre algorithme dans la section II.3.3. Le schéma III.1 rappelle les différentes étapes qui constituent notre algorithme. L'idée générale est d'avoir une première phase d'initialisation qui permet de fournir toutes les connaissances a priori que nous avons sur l'objet, objet qui sera pris au sens très large du terme et son environnement. Il s'agira entre autres des probabilités d'existence de chaque partie constituant cet objet. Une fois cette étape d'initialisation terminée, l'algorithme est itératif et va répéter trois actions principales jusqu'à sa terminaison. Les trois actions principales constituant donc le cœur de l'algorithme sont la sélection, la détection et la mise à jour.

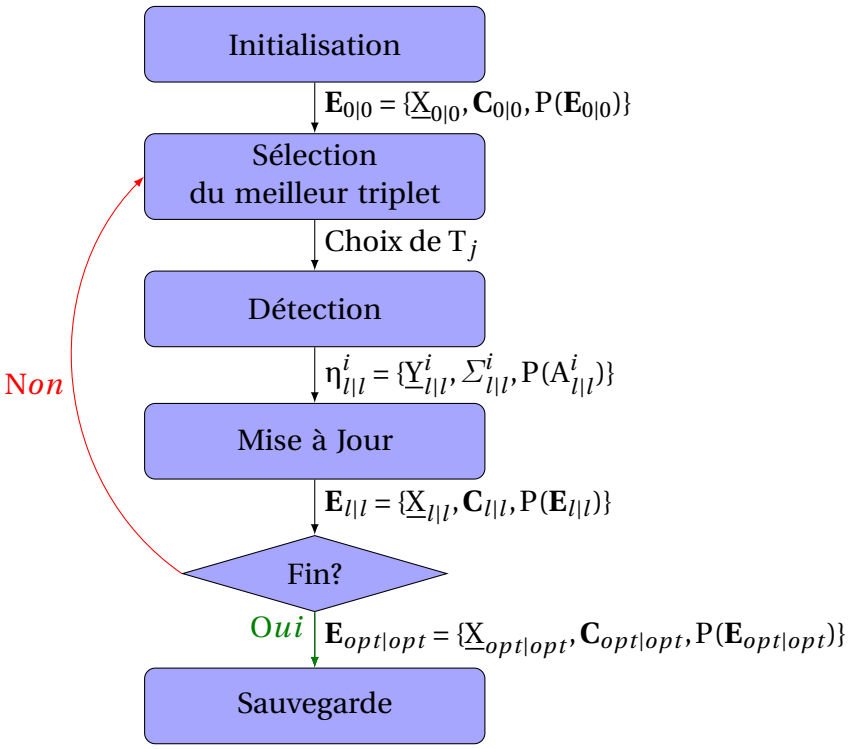


FIGURE III.1 – Schéma simplifié du fonctionnement de notre algorithme avec les grandeurs essentielles après chaque étape.

III.2 Initialisation

III.2.1 Principe

Nous avons vu dans le § II.3.3.1 que lors de cette étape toutes les connaissances a priori sont fournies à l'algorithme. C'est en particulier l'occasion de donner l'état initial de l'objet $\mathbf{E}_{0|0} = \{\underline{X}_{0|0}, \mathbf{C}_{0|0}, P(\mathbf{E}_{0|0})\}$ ainsi que toutes les fonctions $\mathbf{h}_i$ et leurs jacobiennes $\mathbf{H}_i$ liant le vecteur de caractéristiques de l'objet à celui des parties ($\underline{Y}^i = \mathbf{h}_i(\underline{X})$) et les probabilités d'existence $P(A^i_{0|0})$ pour chaque partie i.

III.2.2 Exemple

Pour illustrer ce principe, prenons un exemple simple : nous souhaitons reconnaître un carré dans un espace capteur donné, ici une image. Ce carré sera, de plus, aligné avec les axes. Il sera représenté par le vecteur de caractéristiques suivant : $\underline{X} = \begin{pmatrix} c_u & c_v & c_l \end{pmatrix}^T$ où (c_u, c_v) est le centre du carré dans l'image et c_l la longueur d'un de ses côtés. À l'initialisation, le centre du carré (c_u, c_v) pourra être positionné au centre de l'image. avec une variation acceptée correspondant à la taille de l'image selon chaque axe. Pour la probabilité initiale, il est possible d'affecter à $P(\mathbf{E}_{0|0})$ la probabilité par exemple de 0.5 s'il n'y a pas de connaissance sur la présence ou l'absence du carré dans l'image. Si, à l'inverse, la présence d'un carré dans l'image est attestée, alors $P(\mathbf{E}_{0|0}) = 1$.

$$\mathbf{E}_{0|0} = \{ \underline{X}_{0|0} = \begin{pmatrix} \frac{width}{2} \\ \frac{height}{2} \\ \frac{l_{min}+l_{max}}{2} \end{pmatrix}, \mathbf{C}_{0|0} = \begin{pmatrix} \left(\frac{width}{2}\right)^2 & 0 & 0 \\ 0 & \left(\frac{height}{2}\right)^2 & \\ 0 & 0 & \left(\frac{l_{min}+l_{max}}{2}\right)^2 \end{pmatrix}, P(\mathbf{E}_{0|0}) \}$$

avec

- $height$, la hauteur de l'image
- $width$ la largeur de l'image
- l_{min} et l_{max} tels que la taille du carré soit comprise à un écart-type entre l_{min} et l_{max}

Le carré disposera de quatre parties : *bordGauche*, *bordDroit*, *bordHaut* et *bordBas* nécessitant des primitives de type *ligne*. Chacune des parties pourra, par exemple, être représentée par un vecteur de caractéristiques $\begin{pmatrix} u1_0^i & v1_0^i & u2_0^i & v2_0^i \end{pmatrix}^T$ où $(u1_0^i, v1_0^i)$ et $(u2_0^i, v2_0^i)$ sont les deux points représentant les extrémités de chaque segment. La probabilité d'existence $P(A_{0|0}^i)$ vaudra 1 ce qui signifie que la présence des quatre segments est obligatoire pour constituer un carré. Les fonctions $\mathbf{h}_i$ seront donc les fonctions permettant de calculer à partir du centre du carré et de la taille de celui-ci, la position des coins du carré. D'où :

$$\begin{aligned}
\mathbf{h}_{bordGauche}(\underline{X}) &= \begin{pmatrix} c_u - \frac{l}{2} & c_v - \frac{l}{2} & c_u - \frac{l}{2} & c_v + \frac{l}{2} \end{pmatrix}^T \\
\mathbf{h}_{bordDroit}(\underline{X}) &= \begin{pmatrix} c_u + \frac{l}{2} & c_v - \frac{l}{2} & c_u + \frac{l}{2} & c_v + \frac{l}{2} \end{pmatrix}^T \\
\mathbf{h}_{bordHaut}(\underline{X}) &= \begin{pmatrix} c_u - \frac{l}{2} & c_v + \frac{l}{2} & c_u + \frac{l}{2} & c_v + \frac{l}{2} \end{pmatrix}^T \\
\mathbf{h}_{bordBas}(\underline{X}) &= \begin{pmatrix} c_u - \frac{l}{2} & c_v - \frac{l}{2} & c_u + \frac{l}{2} & c_v - \frac{l}{2} \end{pmatrix}^T
\end{aligned}$$

D'autres caractéristiques nécessaires au bon fonctionnement de l'algorithme seront également initialisées dans cette étape (voir le § III.8) mais pour le moment, nous nous contenterons des paramètres présentés.

Cette étape d'initialisation ne sera faite qu'une seule fois. Une fois les éléments nécessaires initialisés, il est possible de débuter le cœur de l'algorithme. Nous allons donc maintenant parler de la phase de sélection du meilleur triplet.

III.3 Sélection du meilleur triplet perceptif

La figure III.2 récapitule le principe de fonctionnement de cette sélection.

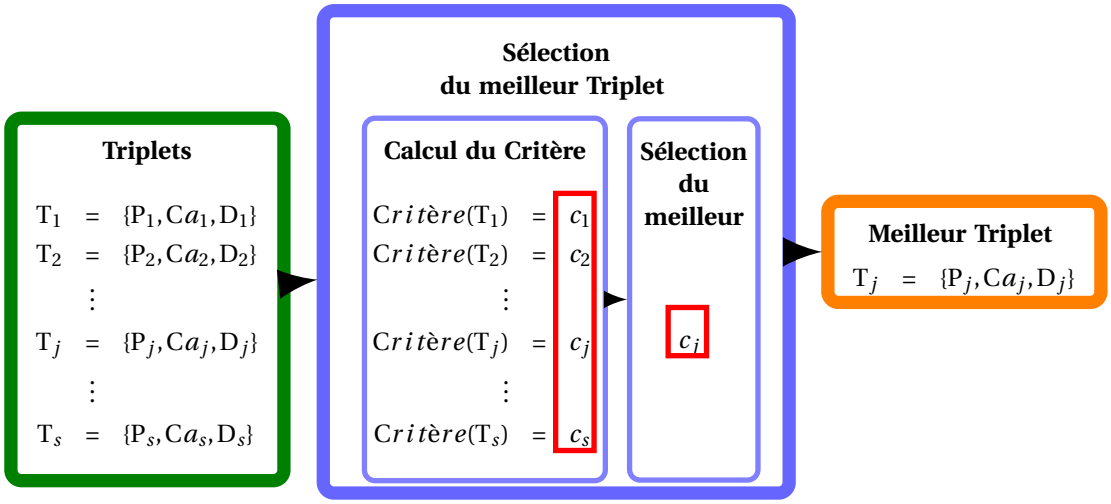


FIGURE III.2 – Schéma de principe de la sélection du meilleur triplet : en entrée, tous les triplets existants sont fournis au sélecteur. Le sélecteur calcule pour chaque triplet un critère. Le triplet ayant le meilleur critère (ici le $j$ième) est sélectionné et utilisé dans les étapes suivantes.

III.3.1 Triplets perceptifs disponibles

Nous avons vu dans le § II.3.3.2, qu'un triplet était constitué d'une partie de l'objet, d'un capteur avec lequel peut être perçue la partie et d'un détecteur, celui-ci étant une brique logicielle conçue pour un but donné. Un récapitulatif des notations pour les triplets est présenté dans la table III.1. Un exemple de triplet peut donc être "un mur" (partie), "un LIDAR" (capteur) et "un algorithme de détection de lignes" (détecteur).

TABLEAU III.1 – Notations pour les triplets perceptifs

j	indice pour les triplets
T_j	triplet perceptif j. $T_j = \{P_j, Ca_j, D_j\}$
P_j	partie du triplet perceptif j
Ca_j	capteur du triplet perceptif j
D_j	détecteur du triplet perceptif j. D_j doit pouvoir permettre de détecter des primitives associées à P_j dans la modalité capteur Ca_j.
s	nombre de triplets possibles
M	nombre de parties constituant l'objet
N_{Ca}	nombre de capteurs disponibles
N_d	nombre de détecteurs disponibles
c_j	critère calculé pour la sélection du meilleur triplet, associé au triplet T_j

Pour une situation donnée, une première étape est donc de constituer la liste des triplets possibles. Si nous appelons M le nombre de parties de l'objet, N_{Ca} le nombre de capteurs disponibles et N_d le nombre de détecteurs à disposition, nous avons donc un

nombre maximum de triplets pouvant atteindre $M \times N_{Ca} \times N_d$. Néanmoins, le nombre réel de triplets est souvent très en deçà de ce maximum. Tout d'abord, toutes les parties ne sont pas perceptibles avec toutes les modalités capteurs, par exemple, "un blob rouge" ne pourra pas être identifié avec un LIDAR. De plus, les détecteurs sont souvent spécifiques et ne peuvent pas être utilisés avec tous les capteurs et pour toutes les parties. Nous noterons s le nombre de triplets possibles.

Enfin, un dernier point qui permet de réduire le nombre de triplets possibles à un instant donné est la focalisation spatiale que nous détaillerons dans le § III.5.1. Prenons l'exemple d'un objet "véhicule" pour une application de type localisation où les différentes "parties" de l'objet seraient en fait les différents amers présents dans une carte géoréférencée (comme c'est le cas dans [Aynaud et al., 2014]). Il n'est pas utile de vouloir considérer un triplet comportant un bâtiment situé à 3 km de la position estimée du véhicule. Les capteurs ne pourront pas réellement le percevoir et donc il ne pourra jamais être détecté. Ce triplet n'est donc pas pertinent. Seuls les triplets qui peuvent avoir une utilité seront sélectionnés. Un triplet sera choisi en fonction notamment de son apport informationnel.

Ainsi, en fonction des capteurs, des algorithmes disponibles et des parties de l'objet, l'ensemble des triplets disponibles peut être construit à chaque instant.

III.3.2 Objectifs du critère de sélection

Dans cette section, l'objectif est de sélectionner de manière astucieuse un triplet. En effet, il ne s'agit pas de choisir au hasard parmi l'ensemble des triplets disponibles mais de prendre en compte nos connaissances actuelles pour déterminer le *meilleur* triplet (c'est à dire le plus prometteur). Ce dernier est le triplet dont on espère qu'il nous amènera le plus rapidement possible vers la réalisation de nos objectifs.

La première étape est donc de définir quels sont ces objectifs. Nous allons les décomposer en deux volets : un objectif de précision et un objectif de confiance, comme nous l'avions déjà mentionné dans le § II.3.2.1.e .

TABLEAU III.2 – Notations pour les objectifs

$\mathbf{C}_{Obj}$	matrice objectif définissant l'incertitude souhaitée en fin d'algorithme.
P_{Obj}	confiance objectif représentant la confiance minimale souhaitée en fin d'algorithme.

III.3.2.1 Objectif de précision

Parmi nos objectifs, celui-ci est sans doute le plus courant. Ainsi pour un algorithme de reconnaissance d'objets dans une image, il pourra s'agir de rechercher à localiser cet objet dans l'image avec une précision de 1, 5, 20 ou plus pixels d'écart par rapport à la *vraie* position de l'objet. Semblablement, lors d'une application de localisation de véhicule, il faut pouvoir rechercher la position du véhicule avec une incertitude, par exemple de 1 cm ou de 1 m.

L'objectif de précision consiste à donner la précision minimale souhaitée lorsque l'algorithme se termine. Cet objectif est grandement dépendant de l'application souhaitée. Une précision de 1 m est insuffisante pour permettre une navigation autonome d'un véhicule dans un environnement fortement contraint, comme une route, mais sera potentiellement suffisante pour une navigation autonome au milieu du désert.

L'utilisateur va donc être chargé de définir les besoins qui lui sont propres en fonction de l'application envisagée. L'objet est modélisé par un vecteur de caractéristiques ayant n composantes. Pour chacune des composantes, une précision souhaitable va être définie. L'objectif de précision va pouvoir être représenté par une matrice objectif $\mathbf{C}_{Obj}$ définissant l'incertitude souhaitée en fin d'algorithme. La plupart du temps, cette matrice sera diagonale et les coefficients diagonaux vaudront le carré de la précision souhaitée pour chacun des paramètres du vecteur de caractéristiques. Pour savoir si l'objectif de précision est atteint, il suffira alors de comparer la matrice objectif $\mathbf{C}_{Obj}$ avec la matrice de covariance $\mathbf{C}_{l|l}$ de l'état courant de l'objet.

Cette modélisation permet de prendre en compte des éléments variables : ainsi il sera possible d'avoir un objectif de 1° sur une composante représentant un angle et un objectif de précision de 3 cm sur une position.

Nous venons donc de présenter le premier élément de nos objectifs : *l'objectif de précision*. Nous allons maintenant discuter du second : *l'objectif de confiance*.

III.3.2.2 Objectif de Confiance

De manière plus originale que l'objectif de précision, nous avons choisi d'avoir également un objectif de confiance. Atteindre l'objectif de précision fixé est certes important mais insuffisant dans certaines situations. Cette précision n'est guère intéressante si elle a peu de chance d'être vraie.

Nous pouvons illustrer cela dans le cas d'une application de localisation. Prenons l'exemple d'un robot localisé à 1 cm devant une porte alors qu'il y en a deux. L'estimation de sa position est donc précise mais ambiguë. La confiance est seulement de 0.5. Donc même si l'objectif de précision est atteint, il faut récolter de nouvelles informations pour confirmer (ou infirmer) le choix effectué. L'objectif de confiance est là pour indiquer qu'une confiance de 0.5 n'est pas suffisante pour confirmer la vraie position. Dans ce cas, l'algorithme doit exploiter d'autres informations pour améliorer cette confiance.

L'objectif de confiance va représenter la confiance minimale qu'on souhaite avoir dans l'estimation finale et sera représenté sous la forme d'une probabilité P_{Obj}. Il sera à comparer à la confiance $P(\mathbf{E}_{l|l})$ issue de l'estimation courante de l'objet. Pour être certain de l'estimation P_{Obj} vaudra donc 1. Néanmoins, avoir une confiance totale et absolue avec des détecteurs qui ne sont pas parfaits est impossible à obtenir. Il peut donc être judicieux de revoir ces attentes à la baisse de manière plus conforme à une réalité. La valeur de P_{Obj} sera fixée par l'utilisateur.

Nos deux objectifs ont été définis. Nous allons construire un critère de sélection des triplets nous permettant le plus probablement de progresser au mieux vers ces objectifs.

III.3.3 Contraintes à respecter pour la sélection du triplet

Pour le calcul du critère de sélection du meilleur triplet, plusieurs contraintes doivent donc être respectées :

— si le triplet n'apporte pas d'information nouvelle, il ne doit pas être sélectionné. Ainsi, un triplet qui a déjà été appelé précédemment dans exactement les mêmes conditions de détection ne devra pas être réutilisé immédiatement.

— si la primitive associée à un triplet a très peu de chances d'être détectée alors le critère associé devra être bas. Si notre détecteur de couleur est mauvais, il est donc peu probable qu'il trouve une primitive de la bonne couleur. Par conséquent, commencer par chercher une partie dont la caractéristique principale est la couleur ne semble pas pertinent.
De plus, si la partie a une probabilité d'existence plus faible qu'une autre, cela veut dire que la primitive associée a également moins de chance d'être trouvée (parce qu'il est possible que la partie ne soit pas présente et qu'en conséquence il n'y ait rien à détecter).
Enfin, cela comprend aussi les problèmes d'occultation, s'il y a de fortes chances qu'une partie ne soit pas observable compte tenu de l'estimation courante, il semble plus judicieux de chercher à détecter une partie plus visible.

— si les risques d'ambiguïtés de détection sur la partie sont élevés, il sera pareillement préférable d'en choisir une autre.

— il est souvent préférable de privilégier un détecteur un peu moins précis mais plus rapide qu'un détecteur plus précis mais plus lent. En effet, le détecteur plus précis pourra être appelé à l'itération suivante. Si la détection avec le détecteur rapide a réussi, l'estimation de l'objet sera plus précise (ne serait-ce qu'un peu) et donc il sera possible d'appeler le détecteur plus lent sur moins de données (par exemple dans une zone plus petite d'une image grâce au principe de focalisation).

— toute chose égale par ailleurs (même chance de réussir la détection, même temps de calcul, etc), si un triplet apporte plus de précision qu'un autre alors son critère devra être plus élevé.

III.3.4 Exemple Illustratif

Pour l'exemple, l'objet cherché est un rectangle ABCD. Cet objet est décomposé en quatre parties : les quatre coins du rectangle. Le seul et unique capteur disponible est une caméra. Le seul détecteur disponible est un détecteur de coins. Au départ, il y a donc seulement quatre triplets disponibles, chacun constitué d'un coin comme partie, de la caméra comme capteur et du détecteur de coin comme détecteur. Le choix du triplet, dans ce cas précis, se résume au choix de la partie (puisque c'est la seule différence entre les triplets).

Lors d'une première étape, le coin A est supposé avoir été détecté. La figure III.3 montre la situation. Les connaissances a priori sur le rectangle sont telles que les variations admises pour la largeur du rectangle soient plus faibles que celles pour la longueur.

Chercher à nouveau le coin A est inutile : le rectangle est statique, il y a un seul capteur et un seul détecteur donc chercher à nouveau cette partie n'apportera aucune information. À l'inverse, si le coin C est trouvé, le rectangle sera complètement déterminé (l'apport en

précision est donc élevé). Par contre, lors du choix d'un coin dans l'ellipse correspondante, il est tout à fait possible de détecter un outlier du fait de la grande taille de l'ellipse d'incertitude. À l'inverse, choisir le coin B ou D est assez sûr. Une des deux coordonnées est fixée par la position de A (l'ordonnée pour B et l'abscisse pour D) et ne pourra donc que très peu varier. Néanmoins, trouver B ou D apportera moins de précision que C : il restera un point à déterminer. Dans les deux cas, il faudra faire une troisième itération. Si C est choisi (un choix "de précision"), à moins d'avoir de la chance, il y aura eu plusieurs candidats possibles et il faudra confirmer le candidat sélectionné. Globalement, le choix va donc se faire entre B et D. La zone de détection pour B est plus large que celle pour D. Le choix final se portera donc sur D. En effet, la zone étant plus petite, il faudra à la fois moins de temps pour le détecteur afin de réaliser sa détection et les chances de trouver dans cette "petite" zone un autre coin ayant la même abscisse que A est faible.

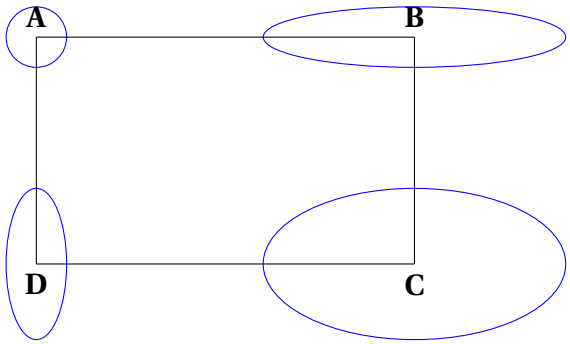


FIGURE III.3 – Un rectangle ABCD est recherché dans une image. À l'étape $l-1$, Le coin A a été détecté (l'imprécision sur sa position est donc faible et liée aux limitations du détecteur). Pour chaque coin, les ellipses bleues représentent les zones de recherche associées pour l'étape l.

Les différentes contraintes pour définir un critère réaliste peuvent être classées en quatre grandes catégories :

— la notion d'information nouvelle nécessaire pour ne pas reprendre toujours la même partie

— la notion de temps du détecteur

— la notion de précision apportée si la détection est réussie

— la notion de probabilité de faire non seulement une détection mais la **bonne** détection : c'est à dire parvenir à détecter mais également à faire la bonne association quand il y a plusieurs candidats.

III.3.5 Contrainte de temps

Nous allons commencer par le temps du détecteur. Il s'agit du temps nécessaire au détecteur pour renvoyer son résultat (la liste de primitives détectées). Ce temps peut être dépendant du nombre de primitives à trouver. Néanmoins, ce nombre n'est pas toujours accessible. Bien qu'il soit possible d'en faire une approximation (voir le § III.4.7.4.a), pour les temps de calculs, nous estimons que le principal facteur est lié à la taille de la région (ROI) dans laquelle le détecteur est exécuté.

Notons t_d le temps du détecteur que nous cherchons à estimer. Nous approximerons t_d par

$$t_d = t_{d_{ref}} \frac{\mathrm{A}_d}{\mathrm{A}_{d_{ref}}} \tag{III.1}$$

où A_d est l'aire de la région à traiter, $t_{d_{ref}}$ est le temps pris par le détecteur pour une région de référence et $\mathrm{A}_{d_{ref}}$ l'aire de cette région. Nous considérons, en effet, que le temps de détection est proportionnel à la taille de la zone d'analyse. $t_{d_{ref}}$ et $\mathrm{A}_{d_{ref}}$ sont supposés connus. Il faut remarquer qu'en particulier pour les détecteurs dans les images, ces données sont souvent fournies et participent à l'estimation et à la comparaison de la qualité du détecteur [Viola and Jones, 2001].

TABLEAU III.3 – Notations pour le temps du détecteur

t_d	temps estimé pour que le détecteur traite les données capteur.
A_d	aire de la région à traiter.
$t_{d_{ref}}$	temps pris par le détecteur pour une région de référence.
$\mathrm{A}_{d_{ref}}$	aire de la région de référence.

L'aire A_d va dépendre de la zone d'intérêt définie à partir de l'estimation courante et du triplet sélectionné. Nous allons donc nous pencher sur le processus de focalisation qui va nous permettre d'obtenir cette région.

III.3.6 Focalisation

Ce phénomène de focalisation avait déjà été mentionné dans le § II.3.3.4. Cette focalisation peut être découpée en deux parties distinctes : une focalisation spatiale et une focalisation de caractéristiques.

III.3.6.1 Focalisation de caractéristiques

Le principe de la *focalisation de caractéristiques* est de déduire, compte tenu de l'estimation à l'étape l de l'objet, des caractéristiques **attendues** pour une partie donnée ce qui permettra d'en déduire les primitives compatibles avec cette partie. Nous y reviendrons dans le § III.5.2.

Selon le type de détecteur, cette *focalisation de caractéristiques* sera utilisée plus ou moins tôt. En effet, il existe deux grands types de détecteurs : les détecteurs paramétrables et les autres. Les premiers vont être paramètrés afin de ne rechercher que des primitives compatibles avec la partie cherchée. Par exemple, si nous disposons d'un détecteur de lignes pour lequel le paramètre caractérisant la pente de la droite peut être précisé et si une ligne horizontale est recherchée alors le détecteur nous renverra uniquement ce qu'il aura détecté comme étant des lignes de pente compatible avec l'intervalle de variance de la pente de la ligne cherchée (c'est à dire des lignes horizontales ou presque). Dans le cas du détecteur de lignes non paramétrable, il renverra toutes les lignes trouvées dans la ROI. Dans l'exemple précédent, comme seules les lignes horizontales nous intéressent, un tri sera effectué sur l'ensemble des lignes renvoyées pour ne conserver que les lignes horizontales. Dans les deux cas, il y a donc un filtrage sur les caractéristiques qui sera fait : soit directement lors de la recherche de primitives, soit via un filtrage une fois la recherche

terminée. Ce filtrage provient de la *focalisation de caractéristiques*.

Cette dernière vient de la manière dont nous avons modélisé les parties dans le § II.3.2.2. Elle est issue de la prédiction de l'état d'une partie (présentée dans le § II.3.2.2.d). L'état prédit correspond à l'état attendu pour une partie. Cette *focalisation de caractéristiques* donne ainsi des valeurs attendues pour chaque caractéristique de la partie.

III.3.6.2 Focalisation spatiale

L'objectif de la *focalisation spatiale* est de déterminer la zone *intéressante* ou région d'intérêt (**ROI**) en fonction de l'état courant estimé de l'objet et de la partie à chercher (voir le § III.5.1). L'idée est de dire que si une partie *bouche* et une partie *front* ont déjà été trouvées pour un objet *visage* dans une image, les yeux sont entre les deux et pas n'importe où dans l'image.

La *focalisation spatiale* va, en fait, découler de la *focalisation de caractéristiques*. Grâce à elle, il est possible de déterminer ce qui est attendu pour la partie cherchée et donc d'en déduire une zone de recherche, qui sera par la suite appelée **zone de focalisation**. Nous reviendrons plus en détails sur cette *focalisation spatiale* dans le § III.5.1.

Nous considérons donc que nous sommes capables de déterminer la zone de focalisation et par conséquent de calculer son aire, ce qui nous permet d'estimer le temps du détecteur (pour un triplet donné puisque la zone sera dépendante de la partie et du capteur). Nous allons maintenant nous intéresser à l'information nouvelle.

III.3.7 Information Nouvelle

L'idée ici est de choisir un triplet selon son contenu informationnel. En effet, dans le cadre d'une application de type localisation, par exemple, si à l'instant t un triplet a été détecté et qu'aucun changement n'intervient, en particulier le véhicule ou robot est immobile, détecter à nouveau ce triplet n'améliorera pas la confiance. Ce n'est pas parce que le même triplet est revu dans des conditions strictement identiques que la confiance dans l'estimation courante pourra augmenter.

TABLEAU III.4 – Notations pour la notion d'information nouvelle

I_l	événement *l'information du triplet* T_j *à l'étape* l *de l'algorithme est indépendante des étapes précédentes.*
$\mathrm{P}(\mathrm{I}_l)$	probabilité de l'événement I_l.
l_{prec}	indice de la précédente itération à laquelle a été sélectionné le triplet T_j (si elle existe).
$\mathrm{D}_{l,l_{prec}}$	distance parcourue entre l'itération l_{prec} et l'itération l.
γ	facteur d'oubli (dépendant de l'application).

De plus, même en terme de précision, reprendre en compte la donnée est, la plupart du temps, un inconvénient plutôt qu'un avantage à cause des problèmes de surconvergence. C'est notamment ce qui peut arriver avec un filtre de Kalman. Les données sont supposées être indépendantes alors que lorsque le même triplet est réutilisé à peu de temps d'intervalle, les données sont très fortement corrélées. L'estimation obtenue risque

alors de devenir fausse. Certaines méthodes [Benaskeur, 2002, Jindal and Psounis, 2004] comme l'intersection de covariance [Simon Julier and JeffreyK Uhlmann, 2001] ont été développées pour répondre à ce problème. Néanmoins ces approches traitent le problème de surconvergence dans le cas continu mais pas en ce qui concerne la confiance, qui ne peut être évaluée que de manière ponctuelle. Il faudrait alors traiter explicitement ces liens de corrélations au niveau des probabilités. Il nous parait plus judicieux d'éviter cette situation en écartant les triplets n'apportant pas d'information nouvelle.

La nouveauté d'une information n'est pas binaire : évidemment, un triplet pour lequel aucune tentative de détection n'a encore été réalisée apportera une information nouvelle mais un triplet qui a déjà été détecté précédemment le peut aussi. Par exemple, lorsque dans une application de localisation d'un véhicule, un triplet est sélectionné en début de parcours pour la détection d'un amer donné, il peut être utile de choisir le même triplet si le véhicule repasse ultérieurement devant le même amer. Nous noterons I_l l'événement : *l'information du triplet* T_j *à l'étape* l *de l'algorithme est indépendante des étapes précédentes*, $\mathrm{D}_{l,l_{prec}}$ la distance parcourue depuis la précédente tentative de détection du triplet T_j. On aura alors :

$$\mathrm{P}(\mathrm{I}_l) = \begin{cases} 1 \text{ si le triplet n'a encore jamais été sélectionné} \\ 1 - \exp^{-\gamma \mathrm{D}_{l,l_{prec}}} \text{ sinon} \end{cases} \tag{III.2}$$

où γ est un facteur d'oubli dont la valeur est dépendante de l'application. Pour des applications de type localisation, il a été fixé avec succès de manière empirique à 0.7 pour une vitesse d'évolution de 5 km/h du véhicule. Pour des applications de suivi d'objets, tant que la même image est traitée, le même triplet ne sera jamais repris (aucune nouvelle information) alors qu'au moindre changement d'images, le triplet pourra être à nouveau sélectionné.

Nous allons nous intéresser maintenant à l'apport en précision espéré pour chacun des triplets potentiels.

III.3.8 Apport en Précision

TABLEAU III.5 – Notations pour l'apport en précision

n	nombre de composantes du vecteur de caractéristiques $\underline{\mathrm{X}}$
$\underline{\mathrm{Y}}^i_{l\|l-1}$	vecteur de caractéristiques prédit pour la partie i=P_j
$\eta^{\mathrm{T}_j}_{l\|l}$	état pour la partie i après une mise à jour utilisant $\underline{\mathrm{Y}}^i_{l\|l-1}$ en guise de mesure.
$\mathbf{E}^{\mathrm{T}_j}_{l\|l}$	état de l'objet mis à jour en utilisant la partie i=P_j ayant pour état $\eta^{\mathrm{T}_j}_{l\|l}$.
$\mathbf{C}^{\mathrm{T}_j}_{l\|l}$	matrice de covariance liée à l'état $\mathbf{E}^{\mathrm{T}_j}_{l\|l}$. Elle représente la précision espérée si la détection liée au triplet T_j réussit.
$\mathbf{C}_{l\|l-1}$	matrice de covariance prédite pour l'itération l. Elle correspond à la précision de l'estimation avant détection (et donc avant mise à jour).
$\mathbf{C}(m,m)$	$m$ième coefficient de la matrice $\mathbf{C}$
$\mathrm{P}^{\mathrm{T}_j}$	apport en précision espéré.

Pour estimer l'apport en précision d'un triplet $T_j = \{P_j, Ca_j, D_j\}$, il faut déterminer quelle serait la précision de l'estimation si la détection (avec le triplet sélectionné) réussissait. Cette précision dépendra partiellement de la primitive compatible sélectionnée (si elle existe). L'objectif, ici, n'est donc pas de connaître exactement la nouvelle précision mais d'en faire une estimation. Pour cela, la partie P_j associée au triplet T_j est mise à jour en utilisant son vecteur de caractéristiques prédit $\underline{Y}^i_{l|l-1}$ en guise de mesure (voir le § II.3.2.2.d pour la prédiction et le § III.5.3 pour le procédé de mise à jour de la partie). Nous notons l'état de la partie ainsi mis à jour : $\eta^{T_j}_{l|l}$. Une mise à jour de l'estimation (temporaire) de l'état de l'objet est également faite (voir le § III.6.2) à l'aide du nouvel état $\eta^{T_j}_{l|l}$ de la partie P_j. Nous obtenons alors un nouvel état pour l'objet, noté $\mathbf{E}^{T_j}_{l|l}$ et en particulier la matrice de covariance $\mathbf{C}^{T_j}_{l|l}$ qui représente donc la précision espérée en cas de réussite de la détection du détecteur D_j pour la partie P_j.

Nous définissons alors :

$$P^{T_j} = \sqrt{\sum_{m=1}^{n} \left(\frac{\mathbf{C}_{l|l-1}(m,m) - \mathbf{C}^{T_j}_{l|l}(m,m)}{\mathbf{C}_{Obj}(m,m)} \right)^2} \tag{III.3}$$

où

- P^{T_j} est l'apport en précision pour le triplet T_j considéré
- $\mathbf{C}(m,m)$ désigne le $m$ième coefficient diagonal de la matrice $\mathbf{C}$
- $\mathbf{C}_{Obj}$ est la matrice objectif qui représente la précision désirée pour chaque paramètre du vecteur de caractéristiques (voir le § III.3.2.1). Elle est, de plus, ici supposée être diagonale mais nous aurions pu considérer le cas général [2].
- $\mathbf{C}_{l|l-1}$ est la matrice de covariance prédite pour l'itération l (voir le § II.3.2.1.f). Elle correspond à la précision de l'estimation avant détection (et donc avant mise à jour).
- $\mathbf{C}^{T_j}_{l|l}$ représente la nouvelle précision espérée.

Avec la formulation de l'équation (III.3), l'objectif de précision est vu comme l'échelle de mesure qui permet de mettre sur le même plan les avancées sur chaque axe et de calculer une distance de progression. Ainsi, si l'objectif est de 0.1 m sur un axe x, de 1 m sur un axe y et de 1° sur un dernier axe θ alors l'apport en précision sera le même pour les trois cas suivants :

- la précision est améliorée de 0.1 m sur l'axe x et de 0 sur les autres.
- la précision est améliorée de 1 m sur l'axe y et de 0 sur les autres.
- la précision est améliorée de 1 ° sur l'axe θ et de 0 sur les autres.

Cet apport en précision présente l'avantage de tenir compte de l'objectif et d'être rapide à calculer. Néanmoins, avec la formulation de l'équation (III.3), deux triplets peuvent avoir le même apport en précision alors que l'un améliore uniquement un paramètre pour lequel l'objectif est atteint tandis que l'autre non. Dans l'exemple précédent, si la précision précédente était de 0.1 sur l'axe x et de 2 sur l'axe y, que le triplet T_1 permet d'améliorer de 0.05 la précision sur l'axe x (et 0 ailleurs) tandis que le triplet T_2 permet d'améliorer de 0.5

la précision sur l'axe y, ils auront le même apport en précision. Pourtant le triplet T_2 est plus intéressant pour progresser vers l'objectif. Il est possible de modifier la formulation originelle afin de favoriser les triplets qui améliorent la précision des paramètres pour lesquels l'objectif n'est pas encore atteint :

$$P^{T_j} = \sqrt{\sum_m (r(m)\, isSup0(r(m)))^2} \tag{III.4}$$

où

$$r(m) = \frac{\mathbf{C}_{l-1|l-1}(m,m) - \max(\mathbf{C}^{T_j}_{l|l}(m,m), \mathbf{C}_{Obj}(m,m))}{\mathbf{C}_{Obj}(m,m)}$$

$$isSup0(x) = \begin{cases} 1 \text{ si } x \geq 0 \\ 0 \text{ sinon} \end{cases}$$

Pour ce calcul, la matrice objectif est supposée diagonale mais pas nécessairement les autres matrices de covariance. En effet, la racine carré des coefficients diagonaux de la matrice de covariance représente la borne maximum (voir annexe A) sur chacun des axes. Elle représente donc l'avancée de cette borne sur l'axe en regard avec notre objectif. La figure III.4 montre un exemple en dimension 2.

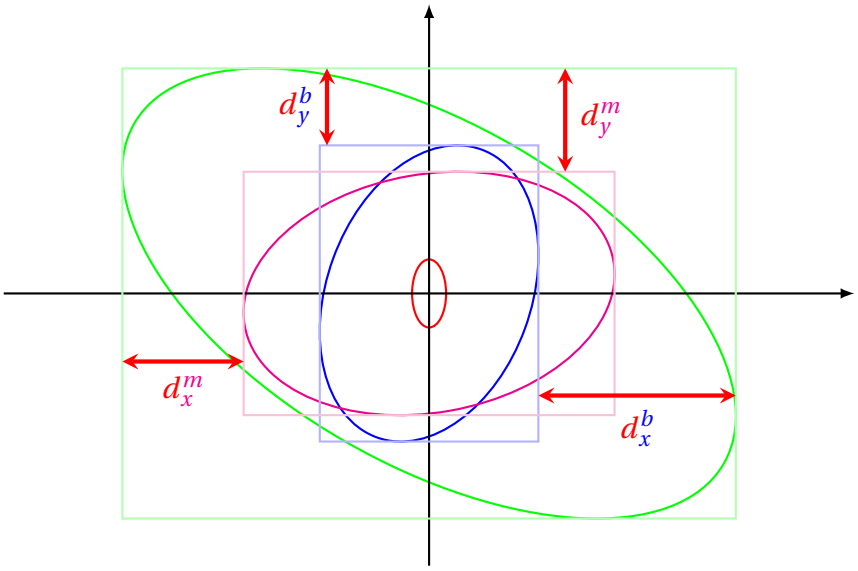


FIGURE III.4 – Principe du calcul de l'apport en précision sur un point. En rouge, l'ellipse représentant l'objectif de précision. En vert, l'ellipse représentant la covariance prédite (et donc la précision avant détection). En bleu et magenta, deux exemples de nouvelles covariances possibles (après détection). Le gain sur chaque axe est représenté par les flèches rouges. L'apport en précision pour chacun des deux exemples pourra ensuite être calculé en fonction de chacune des distances partielles d_x et d_y et de l'objectif de précision sur chacun des axes.

Comme la matrice objectif $\mathbf{C}_{Obj}$ a été supposée diagonale, les axes de la base canonique sont aussi les axes de la matrice objectif. Si la matrice objectif n'est pas diagonale, elle est néanmoins toujours diagonalisable. Donc, il suffirait de calculer les vecteurs propres et les valeurs propres de la matrice objectif, d'en déduire la base représentative de $\mathbf{C}_{Obj}$ et d'effectuer le changement de base correspondant pour toutes les autres matrices.

Nous avons proposé une formulation pour l'apport en précision sous la forme d'une distance parcourue sur chaque axe rapportée à l'objectif défini. Elle permet donc de mettre sur un même plan, grâce à la matrice objectif, des avancées sur des données de nature différente (des mètres ou des degrés par exemple). Il est alors possible grâce à cette distance de déterminer quel triplet semble apporter le plus de précision en fonction de l'objectif de précision souhaité.

Il ne reste donc plus qu'un élément à traiter pour définir enfin le critère de sélection. Il s'agit de la notion de probabilité de réussir une détection et plus encore la **bonne** détection.

III.3.9 Probabilité de bonne détection

TABLEAU III.6 – Notations pour la probabilité de bonne détection

d_{b_l}	événement *une primitive compatible (au moins) a été détectée et l'association partie -primitive est la bonne.*
$\mathrm{P}(d_{b_l})$	probabilité de l'événement d_{b_l}

Dans le § III.3.3 différents événements aléatoires influençant le choix du critère avaient été évoqués. Les différents événements permettant de qualifier une détection seront présentés en détails dans la section III.4. Nous admettons ici que la modélisation des différents événements a été faite et qu'un réseau bayésien est utilisé pour représenter les liens entre ces événements ainsi que pour calculer les probabilités associées.

Pour le critère de sélection, nous avons besoin d'évaluer la probabilité de faire une **bonne** détection : c'est à dire la probabilité que la détection réussisse (une primitive compatible avec la partie cherchée a été trouvée) et que la **bonne association** soit faite (parmi l'ensemble des primitives compatibles, celle qui correspondait réellement à la partie recherchée, a été choisie). Pour avoir la probabilité d'avoir effectué la **bonne** détection, il suffira donc de calculer (à l'aide du réseau bayésien) la valeur $\mathrm{P}(d_{b_l})$ où d_{b_l} est l'événement : *une primitive compatible (au moins) a été détectée et l'association partie-primitive est la bonne* (voir le § III.4.7 pour plus de précisions).

III.3.10 Critère de sélection du meilleur triplet perceptif

Maintenant que nous avons vu les différents éléments que nous voulions prendre en compte, nous pouvons exprimer le critère c_j définissant la pertinence du triplet T_j :

$$c_j = \mathrm{P}(d_{b_l}) \frac{\mathrm{P}^{\mathrm{T}_j}}{t_d} \mathrm{P}(\mathrm{I}_l) \tag{III.5}$$

avec t_d (voir le § III.3.5) le temps estimé que nécessitera le détecteur pour réaliser sa détection (qui dépendra du détecteur mais aussi de la taille de la zone), $\mathrm{P}^{\mathrm{T}_j}$ (voir le § III.3.8) l'apport en précision du triplet (qu'il est possible de voir comme une distance parcourue en fonction des objectifs), $\mathrm{P}(d_{b_l})$ (voir le § III.3.9) la probabilité de faire une bonne détection et $\mathrm{P}(\mathrm{I}_l)$ (voir le § III.3.7) la probabilité que le triplet apporte une information nouvelle.

Cette formulation du critère peut être vue comme une vitesse probable d'avancée vers les objectifs. Le meilleur triplet T_{best} sera alors tel que $best \in [0, s]$ soit l'indice pour lequel

$$c_{best} = \max_j(c_j) \tag{III.6}$$

III.3.11 Quelques remarques sur le critère de sélection

Pour définir le critère de sélection, nous nous sommes basés sur un ensemble de caractéristiques nous semblant importantes à prendre en compte comme le temps nécessaire pour la détection ou l'apport en précision d'un triplet. Nous avons ensuite conçu un critère global à partir de ces caractéristiques.

Néanmoins, il pourrait sembler plus naturel et simple de penser :

a) il suffit de choisir la partie apportant le plus de précision, si la détection réussit, la confiance devrait augmenter naturellement.

b) il suffit de choisir la partie apportant la meilleure confiance, si la détection réussit, la précision augmentera naturellement

Aucune de ces deux affirmations n'est vraie. La proposition a) a le défaut de ne pas tenir compte des problèmes d'associations qui peuvent amener à une diminution importante de la confiance. À l'inverse, la proposition b) pourrait ne pas apporter une progression vers les objectifs de précision. De plus, il est dommage de dire que s'il existe une partie pour laquelle la probabilité de faire une bonne détection est de $\frac{55}{77}$ et une autre pour laquelle cette probabilité est de $\frac{56}{77}$, de toujours choisir la première et ce même si l'apport en précision de la seconde est dix fois plus important : les probabilités de parvenir à la bonne détection sont "presque" identiques, donc autant choisir la deuxième.

Le critère proposé tente de toujours faire un compromis entre le temps pris pour réaliser une détection, la probabilité de la réussir et l'apport en précision que cela générerait.

Nous avons donc vu comment nous pouvions sélectionner le meilleur triplet. Avant d'étudier la phase de détection qui suit la sélection, nous allons présenter de manière détaillée le réseau bayésien que nous utilisons pour calculer les probabilités nécessaires pour notamment estimer ce critère de sélection pour chaque triplet.

III.4 Construction de notre réseau bayésien

L'avantage d'utiliser un réseau bayésien est qu'il permet une visualisation simple et assez intuitive des événements pris en compte : les événements sont modélisés par des nœuds et les causes sont reliées au effet par des flèches. Il offre également l'avantage de pouvoir faire de l'**inférence bayésienne** [1] : les différentes observations peuvent être prises en compte et les probabilités, liées à chaque événement, recalculées. Le réseau bayésien est donc particulièrement adapté dans notre cas puisqu'il permet à la fois de gérer des événements incertains mais également de recalculer au fur et à mesure les probabilités des événements en fonction des nouvelles données.

Nous allons décrire notre réseau bayésien étape par étape en partant d'une représentation volontairement simple qu'on complexifiera au fur et à mesure. Les prérequis sur les réseaux bayésiens et les probabilités sont présentés en annexe C. L'avantage de faire ainsi

1. procédé par lequel il est possible de prendre en compte la ou les connaissances a posteriori (après observation) sur certains événements (on **sait** s'ils se sont réalisés ou non) et de voir l'influence de ces connaissances sur la probabilité des autres événements

plutôt que de présenter directement le réseau final est de permettre de se familiariser avec les différents événements et de bien voir les liens qu'ils ont entre eux.

III.4.1 Première modélisation

Pour notre première modélisation, nous avons un monde parfait et idéal. Dans ce cas, les événements considérés sont : l'**intégrité**[2] de l'estimation courante, la réussite (ou l'échec) de la détection et l'intégrité de l'estimation après mise à jour. Le procédé de mise à jour sera décrit ultérieurement (voir le § III.6.2). Ces événements sont notés :

— $O_{l|l-1}$: l'estimation avant mise à jour, définie à l'aide de $\underline{X}_{l|l-1}$ et $\mathbf{C}_{l|l-1}$, est intègre.

— d_l : la détection à l'itération l a réussi.

— $O_{l|l}$: l'estimation après mise à jour (c'est à dire après avoir tenté de faire une détection), définie à l'aide de $\underline{X}_{l|l}$ et $\mathbf{C}_{l|l}$, est intègre.

Le réseau bayésien de la figure III.5, avec les tables de probabilités conditionnelles associées (tableau III.7), peut alors être construit :

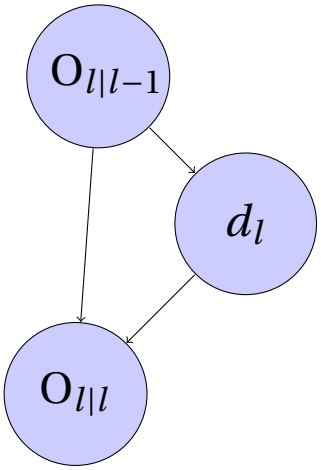


FIGURE III.5 – Premier réseau bayésien. Les événements pris en compte sont les suivants :

— $O_{l|l-1}$: l'estimation avant mise à jour, définie à l'aide de $\underline{X}_{l|l-1}$ et $\mathbf{C}_{l|l-1}$, est intègre.

— d_l : la détection à l'itération l a réussi. d_l est un événement observable.

— $O_{l|l}$: l'estimation après mise à jour, définie à l'aide de $\underline{X}_{l|l}$ et $\mathbf{C}_{l|l}$, est intègre.

TABLEAU III.7 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.5

$O_{l\|l-1}$	d_l
0	0
1	1

a

$O_{l\|l-1}$	d_l	$O_{l\|l}$
0	0	0
0	1	#
1	0	#
1	1	1

b

Observons, un peu plus en détail, les tables de probabilités conditionnelles écrites en III.7. Le monde étant idéal, lorsque l'estimation est intègre, $O_{l|l-1} = 1$, la détection doit être réussie à coup sûr (deuxième ligne de la table III.7a), il n'y a aucun risque de confusion possible. À l'inverse, si l'estimation n'est pas intègre, la détection échoue systématiquement (première ligne de la table III.7a). La première ligne de la table III.7b indique que si l'estimation n'était pas intègre $O_{l|l-1} = 0$, que la détection échoue alors l'estimation après mise à jour ne sera pas intègre non plus : $O_{l|l} = 0$. À l'inverse (voir dernière ligne de la table III.7b), si l'estimation était intègre et que la détection réussit alors la nouvelle estimation

2. L'intégrité de l'estimation représente le fait que le vecteur de caractéristiques $\underline{X}$ à l'étape l est bien compris dans l'ellipse définie à l'aide de l'estimation $\underline{X}_{l|l}$ et de sa covariance associée $\mathbf{C}_{l|l}$.

sera elle aussi intègre. En effet, le processus de mise à jour est supposé cohérent et doit donc conserver l'intégrité lorsque la détection est **bonne**.

Les valeurs marquées par #, deuxième et troisième ligne de la table III.7b, sont celles qui correspondent à des impossibilités dans le modèle. En effet, pour le moment, une détection ne peut pas avoir réussi si l'estimation n'était pas intègre (deuxième ligne de la table III.7a). De la même façon si l'estimation est initialement intègre, dans un monde parfait, le détecteur retournera obligatoirement un résultat. Ces combinaisons d'événements ne peuvent donc pas se produire.

Évidemment, le problème est ici extrêmement simplifié. Une première idée est déjà de penser qu'il est assez probable (en particulier au début quand l'estimation est peu précise) d'avoir plusieurs choix possibles. Cette notion va donc être ajoutée dans le prochain réseau bayésien.

III.4.2 Ajout de la notion d'association

Afin de prendre en compte le fait que le processus d'association n'est pas parfait, il est nécessaire de rajouter, dans le réseau bayésien, l'événement d_{b_l} : *la détection est la **bonne** détection, c'est à dire que le "bon" choix a été fait parmi toutes les primitives candidates possibles* (voir la figure III.6). d_{b_l} correspond donc aussi à l'événement : *une primitive compatible (au moins) a été détectée et l'association partie-primitive est la bonne.*

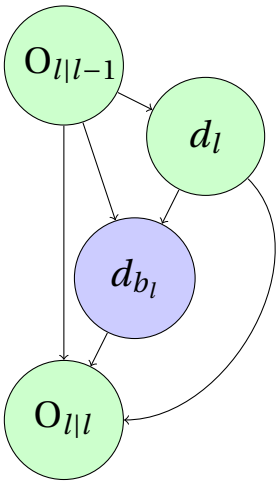


FIGURE III.6 – Réseau bayésien prenant en compte les problèmes d'association. Le nouvel événement est :

— d_{b_l} : la primitive détectée correspond bien à la partie recherchée.

TABLEAU III.8 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.6

$P(\mathbf{E}_{l\|l-1})$	d_l	d_{b_l}
0	0	0
0	1	#
1	0	#
1	1	$\frac{1}{1+\textbf{nbVoisins}}$

a

$O_{l\|l-1}$	d_l	d_{b_l}	$O_{l\|l}$
0	Φ	Φ	0
1	0	Φ	#
1	1	0	0
1	1	1	1

b

Pour faciliter la visualisation, les événements déjà pris en compte précédemment sont affichés en vert (en bleus pour les nouveaux événements). Les valeurs particulières seront mise en exergue dans les tables. Il s'agira :

— de données à fournir à l'algorithme. Elles seront écrites en vert.

— de valeurs à calculer. Elle seront écrites en bleu.

Pour alléger les notations, Φ désigne $vrai$ ou $faux$ quand cette valeur n'influe pas sur le résultat. Par exemple si l'estimation n'était pas intègre ($P(O_{l|l-1}) = 0$) alors elle ne le devien-

dra pas après mise à jour ($P(O_{l|l}) = 0$) et ce peu importe si la détection a réussi ou non. Les tables de probabilités conditionnelles qui n'ont pas été impactées par la modification de modélisation ne seront pas réécrites.

Dans la table III.8a, la dernière ligne indique que si la détection a réussi, il y a $\frac{1}{1+\textbf{nbVoisins}}$ chance d'avoir sélectionné la bonne primitive . Ainsi, **nbVoisins** désigne naturellement le nombre des autres primitives candidates.

La table III.8b, dernière ligne, indique que pour que l'estimation reste intègre, il faut que la **bonne** association ait été faite ($d_{b_l} = 1$). À l'inverse, si le mauvais choix a été fait, troisième ligne de la table III.8b, l'intégrité de l'estimation sera perdue. Ainsi par exemple si pour un carré droit, la primitive associée à la partie *bordGauche* est choisie pour la partie *bordDroit*, l'estimation du carré après mise à jour sera erronée (le carré estimé sera en décalage avec la réalité).

Les erreurs d'association ont donc été prises en compte. Il est nécessaire à présent de prendre en compte les erreurs de détection.

III.4.3 Ajout de la notion d'imperfection des détecteurs

Usuellement, les détecteurs ne sont pas parfaits. Il peut arriver que le détecteur ne renvoie pas la primitive associée à la partie alors qu'elle est belle et bien présente dans la zone de focalisation. À l'inverse, le détecteur peut inventer des primitives : il n'y a rien qui corresponde réellement à ce qu'il est censé détecter et pourtant il renvoie une détection. Pour compléter notre réseau, deux probabilités et un événement da_l sont nécessaires :

— α : probabilité de ne pas détecter une primitive alors qu'elle est présente. α est souvent appelé **taux de faux négatifs**

— β : probabilité de renvoyer une primitive alors qu'il n'y a rien. β est souvent appelé **taux de faux positifs**.

— da_l : la primitive correspondant à la partie recherchée est détectée

Dans la table III.9c, dernière ligne, si la bonne primitive a été détectée alors de manière évidente, une primitive au moins a été détectée. Dans les autres cas, il faut déterminer quelle est la probabilité d'avoir fait une détection (autre que la bonne primitive).

— Premier cas : l'estimation est intègre (troisième ligne de la table III.9c). Il y a donc **nbVoisins** primitives, autres que celle cherchée, dans la zone de focalisation. Nous voulons calculer la probabilité que le détecteur détecte quelque chose, donc qu'il réussisse au moins une détection. Il est en fait plus facile de travailler sur le complémentaire : la probabilité que le détecteur ne fasse aucune détection. Il s'agit donc de la probabilité qu'il ne voit aucune des primitives présentes dans la zone. Pour chaque primitive, la probabilité qu'elle ne soit pas détectée est de α. La probabilité qu'aucune ne soit détectée est donc de $\alpha^{\textbf{nbVoisins}}$. Pour ne rien détecter, il ne faut pas non plus que le détecteur invente une primitive , cet événement a une probabilité $1-\beta$. Au final pour que le détecteur ne renvoie aucune primitive il faut qu'il n'ait détecté aucune des primitives présentes **ET** qu'il n'ait rien inventé : ce qui donne

donc la probabilité $\alpha^{\mathbf{nbVoisins}}(1-\beta)$ [3]. La probabilité de l'événement *avoir détecté au moins une primitive* est donc le complémentaire : $1-\alpha^{\mathbf{nbVoisins}}(1-\beta)$.

— Second cas : l'estimation n'est pas intègre (première ligne de la table III.9c). S'il n'est, dans ce cas, pas très surprenant de ne pas avoir détecté la bonne primitive , cela ne signifie pas pour autant qu'il n'y aura aucune primitive compatible avec la partie. Par exemple, en forêt avec un détecteur d'arbres, il y a de fortes chances qu'à n'importe quel endroit, un arbre puisse être détecté. Il faut donc, de manière générale, estimer le nombre de primitives compatibles en dehors de la zone de focalisation (le calcul est présenté dans le § III.4.7.4). Ce nombre est noté **nbAmerProbable**. Ensuite, le même raisonnement que précédemment est utilisé pour exprimer la probabilité de détecter au moins une primitive compatible dans ce cas.

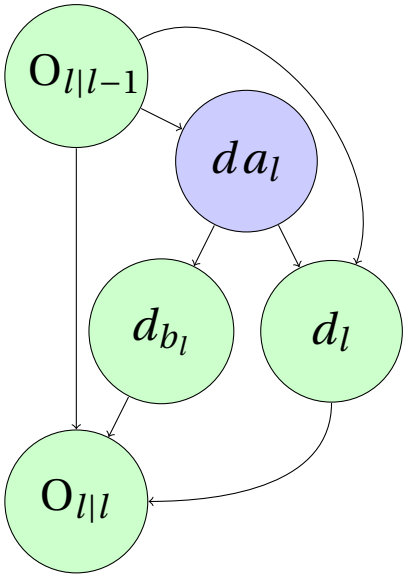


FIGURE III.7 – Réseau bayésien prenant en compte l'imperfection des détecteurs. Le nouvel événement est :

— da_l : la primitive correspondant à la partie recherchée est détectée

TABLEAU III.9 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.7

$O_{l\|l-1}$	da_l
0	0
1	$1-\alpha$

a

da_l	d_{b_l}
0	0
1	$\frac{1}{1+\mathbf{nbVoisins}}$

b

$O_{l\|l-1}$	da_l	d_l
0	0	$1-\alpha^{\mathbf{nbAmerProbable}}(1-\beta)$
0	1	#
1	0	$1-\alpha^{\mathbf{nbVoisins}}(1-\beta)$
1	1	1

c

$O_{l\|l-1}$	d_l	d_{b_l}	$O_{l\|l}$
0	Φ	Φ	0
1	0	Φ	1
1	1	0	0
1	1	1	1

d

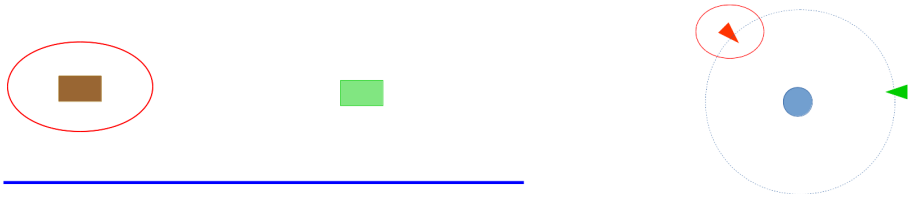
Une des différences entre **nbAmerProbable** et **nbVoisins** est que le premier restera toujours une estimation tandis que le second peut avoir une valeur exacte (par exemple s'il existe une carte de l'environnement).

La table de probabilités III.9d, reste quant à elle, relativement simple à comprendre. La première ligne indique que si l'estimation n'était pas intègre, peu importe les détections et les associations, elle n'a aucune raison de le devenir. À l'inverse, si l'estimation était intègre et que la bonne détection a été faite (dernière ligne) alors l'estimation reste intègre. La deuxième ligne indique que si l'estimation était intègre et qu'il n'y a pas eu de détection alors l'estimation reste intègre. En effet, la mise à jour de l'estimation est impossible (par

3. Les événements *le détecteur ne renvoie pas une primitive alors qu'elle existe* et *le détecteur invente une primitive* sont supposés être indépendants.

contre, la confiance sera modifiée) et donc l'intégrité est conservée.

Les tables de probabilités conditionnelles présentées font l'hypothèse qu'il n'est pas possible de détecter la bonne primitive si l'estimation n'est pas intègre. Or ce n'est pas toujours nécessairement le cas. La figure III.8 montre quelques exemples de situations où il est possible que cela se produise.



(a) Exemple avec un mur (b) Exemple avec un poteau

FIGURE III.8 – Exemple de situations de détection possible même si l'estimation est mauvaise. Sur l'image III.8a un véhicule, dont la vraie position est représentée par un rectangle vert, a sa position estimée représentée par le rectangle marron avec l'ellipse d'incertitude rouge. Pourtant si le mur bleu est cherché, il est possible de le trouver, y compris dans la position erronée. L'idée est similaire pour l'image III.8b, un poteau (représenté en bleu) est cherché à une distance donnée par rapport à un véhicule orienté (représenté par un triangle). Le poteau pourra être détecté à partir de la "vraie" position (en vert) mais également à partir de la position estimée (en rouge).

Pour pallier ce problème, il suffit de modifier la table de probabilité III.9a précédente en définissant :

— ϵ : probabilité de détecter la bonne primitive alors que la position n'est pas intègre.

TABLEAU III.10 – Table de probabilités conditionnelles modifiée pour le réseau bayésien de la figure III.7

$O_{l\|l-1}$	da_l
0	ϵ
1	$1-\alpha$

III.4.4 Ajout de la notion d'observabilité

Il est possible que bien que, l'estimation soit intègre, la partie recherchée ne soit pas visible. Pour modéliser ce problème, nous introduisons la notion d'observabilité de la partie. L'observabilité va permettre de gérer partiellement les problèmes d'occultation car l'observabilité d'une partie dépendra entre autres de si la partie est ou non occultée. Mais le problème de l'observabilité est également lié à l'incertitude sur l'estimation : selon la valeur de cette dernière, une partie peut ou non être observable simplement parce qu'elle sort de l'image capteur.

Prenons, par exemple, le cas d'une caméra qui renvoie une image. L'incertitude sur l'estimation de l'objet va se traduire par une incertitude sur l'estimation des parties. Supposons vouloir trouver trois points A, B et C tels que présentés dans la figure III.9. La zone observable par le capteur (image) est dessinée en blanc. Le gris représente ce qui est en dehors de l'image et donc non observable par ce capteur. Le point A sera toujours observable. Le point C lui au contraire ne le sera jamais. Le point B a une chance sur deux de pouvoir être observé (selon à quelle moitié de l'ellipse appartient le point réel).

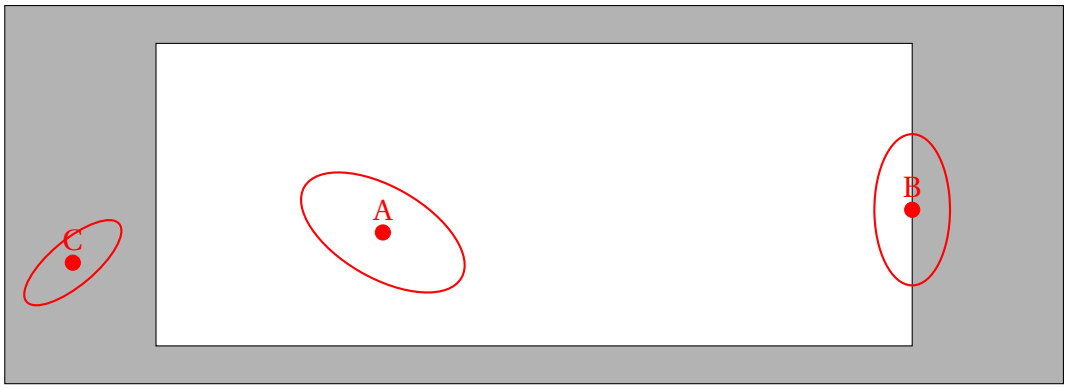


FIGURE III.9 – Observabilité pour 3 points. En blanc, la zone où il est possible d'observer. En gris, une zone où les observations sont impossibles. Les points cherchés sont en rouge.

Notre réseau bayésien (voir la figure III.10) est complété en rajoutant l'événement :

— z_l : la partie recherchée est observable (par le capteur et le détecteur du triplet choisi).

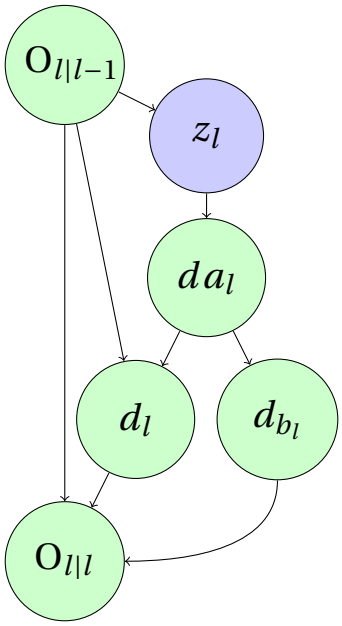


FIGURE III.10 – Réseau bayésien prenant en compte observabilité de la partie. Le nouvel événement considéré est :

— z_l : la partie recherchée est observable (par le capteur et le détecteur du triplet choisi).

TABLEAU III.11 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.10

$O_{l\|l-1}$	z_l
0	ϵ
1	obs

a

z_l	da_l
0	0
1	$1-\alpha$

b

Dans la première ligne de la table III.11a, le fait qu'une partie puisse être observable, même si l'estimation n'était pas intègre, a été conservé, au niveau du noeud z_l. En conséquence, l'événement da_l (la partie recherchée est détectée) ne peut se produire que si la partie était observable (même si son observabilité n'était pas vraiment prévue).

Notons que, dans le cas d'une localisation par carte, avec des amers géoréférencés, nous pourrons estimer facilement la probabilité d'occultation d'un amer par un autre (par exemple, un mur cache un poteau) et nous pourrons l'intégrer dans le réseau.

III.4.5 Ajout de la notion d'existence d'une partie

Certaines parties peuvent ne pas toujours exister : par exemple une partie correspondant à des paires de lunettes. Tous les visages n'auront pas de paires de lunettes. Donc même si la partie est théoriquement dans une zone observable (comme la zone blanche dans la figure III.9), et même avec un détecteur absolument parfait ($\alpha = 0$ et $\beta = 0$), elle pourra ne pas être renvoyée par le détecteur. Le réseau bayésien de la figure III.11 est obtenu en ajoutant l'événement :

— $A^i_{l|l-1}$: la partie i existe.

Dans le tableau III.12, la première ligne représente simplement le fait que si la partie est inexistante, elle ne pourra jamais être détectée. Les deux autres lignes reprennent les principes énoncés précédemment sur l'observabilité.

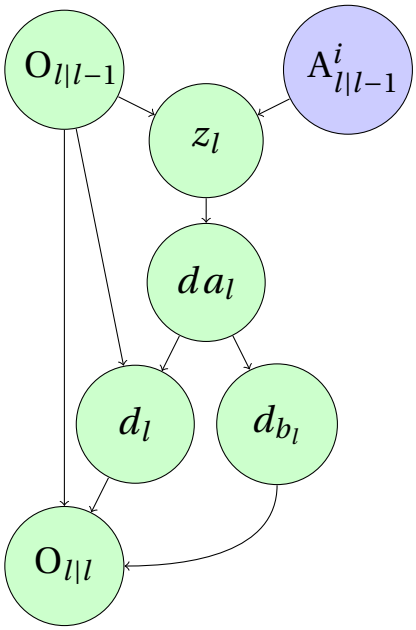


FIGURE III.11 – Réseau bayésien prenant en compte l'existence de la partie. Le nouvel événement ajouté est :

— $A^i_{l|l-1}$: la partie i existe.

TABLEAU III.12 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.11

$A^i_{l\|l-1}$	$O_{l\|l-1}$	z_l
0	Φ	0
1	0	ϵ
1	1	obs

III.4.6 Ajout de la notion d'information nouvelle

Parmi les différents éléments à prendre en compte, il ne reste plus que la notion de nouvelle information comme vu dans le § III.3.7. Ainsi, il est judicieux de considérer que si la situation est exactement la même (même triplet sélectionné dans exactement les mêmes conditions) alors le résultat sera exactement le même et sera donc totalement prévisible. Il devrait également être sans incidence sur la confiance. Si la détection avait déjà échouée, il ne sera pas surprenant qu'elle échoue à nouveau (puisque rien n'a changé) et donc la confiance ne devrait pas diminuer à nouveau. À l'inverse, la réussite d'une détection, dans des conditions identiques à celles d'une précédente détection réussie, ne devrait pas augmenter la confiance. Pour prendre en compte cette notion d'information, un nouvel événement est ajouté à notre réseau (voir figure III.12) :

— I_l : l'information est nouvelle.

En toute rigueur, cette information devrait provenir du réseau de l'étape l_{prec}, où l_{prec} désigne la précédente itération à laquelle le triplet avait été sélectionné, ce qui conduirait à connecter les réseaux de l'itération l et l_{prec}. Cette solution est envisageable mais conduirait à une difficulté calculatoire.

La première ligne du tableau III.13 indique simplement que si l'information n'est pas du tout nouvelle alors le résultat de la détection est prévisible et le résultat rendu sera le même que celui obtenu lors du dernier appel du triplet. Ce résultat précédent de détection est noté $\mathbf{d_{prec}}$. Pour rappel, s'il n'y a pas eu de détection du triplet auparavant alors l'information est considérée comme nécessairement nouvelle (voir le § III.3.7). Les autres lignes viennent directement de la table III.9c (voir le § III.4.3).

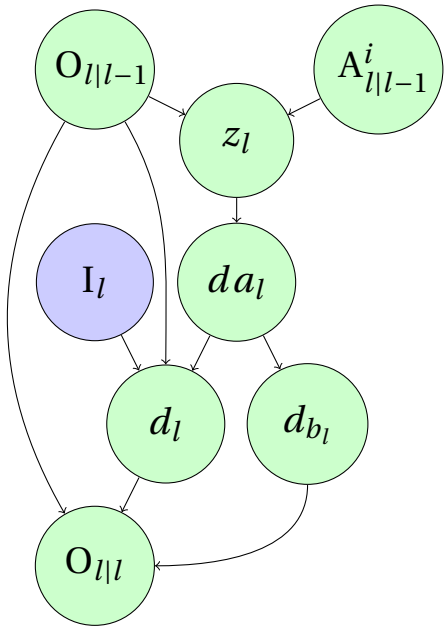


FIGURE III.12 – Réseau bayésien prenant en compte la notion d'information nouvelle. L'événement ajouté est :

— I_l : l'information est nouvelle.

TABLEAU III.13 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.12

I_l	O_{l^-}	da_l	d_l
0	Φ	Φ	$\mathbf{d_{prec}}$
1	0	0	$1-\alpha^{\mathbf{nbAmerProbable}}(1-\beta)$
1	0	1	#
1	1	0	$1-\alpha^{\mathbf{nbVoisins}}(1-\beta)$
1	1	1	1

III.4.7 Réseau Bayésien Final

Nous allons maintenant récapituler les différents événements pris en compte dans notre réseau bayésien ainsi que les différents éléments nécessaires à sa construction. Nous présenterons le réseau complet en mettant en exergue les nœuds d'entrée (leur probabilité est donnée à chaque itération), les nœuds de sortie (ceux dont la valeur nous intéresse) et le nœud d'observation (celui pour lequel une nouvelle donnée est disponible). Nous récapitulerons également toutes les tables de probabilités conditionnelles.

III.4.7.1 Liste des événements

— $O_{l|l-1}$: l'estimation avant mise à jour, définie à l'aide de $\underline{X}_{l|l-1}$ et $\mathbf{C}_{l|l-1}$, est intègre. Il s'agit donc de la confiance qu'on avait avant la détection.

— $A^i_{l|l-1}$: la partie recherchée existe.

— z_l : la partie recherchée est observable dans la modalité capteur choisie.

— da_l : la primitive associée à la partie recherchée a été détectée.

— d_l : au moins une primitive compatible avec la partie a été trouvée (détection réussie).

— d_{b_l} : une primitive (au moins) a été détectée et l'association partie-primitive est la bonne.

— I_l : l'information du triplet est nouvelle (voir le § III.3.7).

— $O_{l|l}$: l'estimation après la mise à jour, définie à l'aide de $\underline{X}_{l|l}$ et $\mathbf{C}_{l|l}$, est intègre. Il s'agit donc de la confiance qu'on a après la tentative de détection.

III.4.7.2 Liste des variables et paramètres nécessaires à la construction des tables

III.4.7.2.a Liste des variables connues

Parmi l'ensemble des variables nécessaires à la construction des tables, plusieurs sont supposées fixées et connues. Il s'agit par exemple des caractéristiques des détecteurs. Dans les tables de probabilités du tableau III.14, il s'agit de toutes les variables apparaissant en vert.

— α : probabilité de ne pas détecter une primitive alors qu'elle est présente.

— β : probabilité de renvoyer une primitive alors qu'il n'y a rien.

— ϵ : probabilité de détecter la bonne primitive alors que la position n'est pas intègre.

Ces paramètres peuvent, par exemple, être appris statistiquement lorsqu'une vérité terrain est disponible.

III.4.7.2.b Liste des variables à calculer

Les variables en bleu dans les tables de probabilités du tableau III.14 sont les variables dont les valeurs vont changer en fonction des itérations, de la partie cherchée et de la zone de focalisation. Il sera donc nécessaire de les calculer à chaque itération de l'algorithme.

— **nbVoisins** : nombre de primitives compatibles avec la partie cherchée, sans compter la bonne primitive (celle qui correspond réellement à la partie), présentes dans la zone de focalisation.

— **nbAmerProbable** : nombre de primitives compatibles probablement présentes dans une zone de taille identique à la zone de focalisation.

— **obs** : probabilité que la primitive présente soit observable lorsque l'estimation est intègre et la partie existante.

— $\mathbf{d_{prec}}$: résultat de la détection précédente du triplet (si précédente détection il y a eu).

III.4.7.3 Graphe et tables de probabilités conditionnelles du réseau bayésien final

TABLEAU III.14 – Tables de probabilités conditionnelles pour le réseau bayésien de la figure III.13

$A^i_{l\|l-1}$	$O_{l\|l-1}$	z_l
0	Φ	0
1	0	ϵ
1	1	**obs**

z_l	da_l
0	0
1	$1-\alpha$

da_l	d_{b_l}
0	0
1	$\frac{1}{1+\textbf{nbVoisins}}$

I_l	$O_{l\|l-1}$	da_l	d_l
0	Φ	Φ	$\textbf{d}_{\textbf{prec}}$
1	0	0	$1-\alpha^{\textbf{nbAmerProbable}}(1-\beta)$
1	0	1	#
1	1	0	$1-\alpha^{\textbf{nbVois}}(1-\beta)$
1	1	1	1

$O_{l\|l-1}$	d_l	d_{b_l}	$O_{l\|l-1}$
0	Φ	Φ	0
1	0	Φ	1
1	1	0	0
1	1	1	1

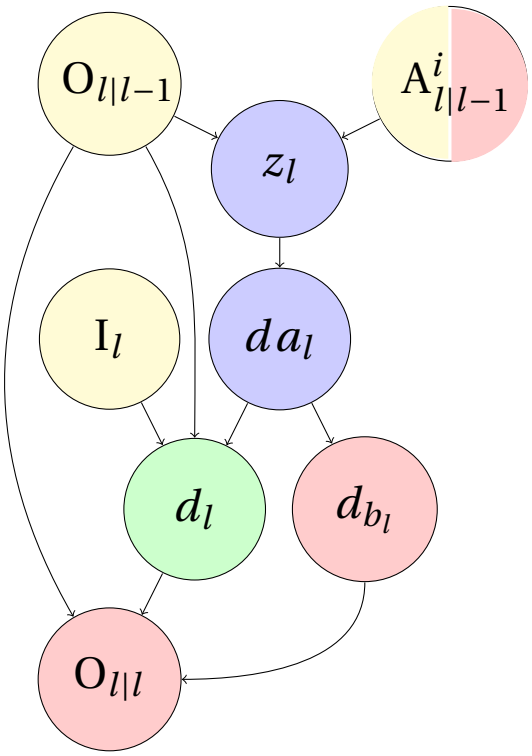


FIGURE III.13 – Réseau bayésien final. En jaune, les nœuds d'entrée, en rose les nœuds dont la probabilité nous intéressera, en vert le nœud d'inférence (celui qui sera observable et qui pourra donc être une évidence) et en bleu les nœuds internes.

III.4.7.4 Calculs des éléments nécessaires

Pour calculer les différentes probabilités de ce réseau bayésien, nous avons donc besoin de préciser comment calculer les valeurs variables dans les tables de probabilités conditionnelles. Nous avons récapitulé la liste des valeurs nécessaires dans le § III.4.7.2. Dans ce paragraphe, nous allons expliquer comment ces valeurs sont calculées.

TABLEAU III.15 – Notations pour les valeurs variables de notre réseau bayésien

nbAmerProbable	nombre probable de primitives compatibles dans la zone de focalisation
Dz_p	densité de primitives du même type que celle cherchée.
A_d	aire de la zone de focalisation
nbVoisins	nombre de primitives compatibles autres que celle cherchée présentes dans la zone de focalisation.
obs	probabilité que la partie soit observable compte tenu de l'estimation courante du vecteur de caractéristiques de l'objet et du capteur utilisé pour la percevoir.
$aire\mathrm{D}ans\mathrm{C}apteur$	aire de la région de la zone de focalisation qui est visible par le capteur
$\mathbf{d_{prec}}$	résultat de la détection précédente (qui a eu lieu à l'itération l_{prec}) si tant est qu'elle existe.

Le nombre probable de primitives compatibles va dépendre de la taille de la zone de focalisation et de la densité de primitives du même type que celle cherchée :

$$\textbf{nbAmerProbable} = Dz_p \times A_d \tag{III.7}$$

où

— Dz_p est la densité de primitives du même type que celle cherchée

— A_d est l'aire de la zone de focalisation.

La densité Dz_p est supposée connue. Par exemple, pour une partie *bordGauche* d'un carré cherché dans une image, il faudra donner la densité de lignes par pixel carré. L'unité de la densité est l'unité du capteur. Ainsi, pour une partie cherchée avec un lidar dans une zone en mètre carré, la densité (par exemple la densité d'arbres) sera une densité par mètre carré.

L'équation (III.7) donne uniquement une estimation. Ainsi si un arbre est recherché dans une forêt, cette densité nous permettra de savoir qu'il est fort probable qu'il y ait d'autres arbres à détecter, y compris si l'estimation n'est pas intègre. Il faut noter que cette estimation est plutôt défavorable : la densité par unité carré est considérée alors que, s'il peut y avoir beaucoup d'arbres dans une forêt, il y en aura déjà moins qui auront un tronc d'une taille précise (*focalisation de caractéristiques*).

III.4.7.4.a Calcul de nbVoisins

Pour le calcul du nombre de voisins, il y a plusieurs cas possibles :

a) des informations précises sur l'environnement sont disponibles.

b) aucune information précise sur l'environnement n'est disponible.

Dans le cas a), par exemple, pour une application de type localisation avec une carte, il suffira simplement de dénombrer les primitives de même type présentes dans la zone de focalisation.

Dans le cas b), par exemple, s'il faut reconnaître un objet dans une image, l'environnement exact n'est pas connu à l'avance et donc il est impossible de faire un simple dénombrement. Dans ce cas là, l'approximation suivante sera utilisée :

$$\textbf{nbVoisins} = \max(0, \textbf{nbAmerProbable} - 1) \tag{III.8}$$

où **nbAmerProbable** est le nombre probable de primitives présentes dans la zone..

III.4.7.4.b Calcul d'obs

Le calcul de l'observabilité d'une partie dans un certain capteur est grandement dépendant des connaissances que nous avons : est-ce que nous pouvons savoir si la partie est occultée ou non ? Y-a-t-il d'autres facteurs que l'occultation qui peuvent gêner la visibilité de la partie et si oui lesquels ?

Selon l'application et les capteurs, un calcul approximatif simple peut être fait. Par exemple pour de la reconnaissance d'objets pour laquelle aucune information annexe n'est disponible, il est possible d'utiliser l'estimation suivante :

$$\textbf{obs} = \frac{aire\mathrm{D}ans\mathrm{C}apteur}{\mathrm{A}_d} \tag{III.9}$$

où

- A_d est l'aire de la zone de focalisation
- $aire\mathrm{D}ans\mathrm{C}apteur$ est l'aire de la région de la zone de focalisation qui est visible par le capteur (voir la figure III.9).

Des calculs plus complexes sont possibles dès lors que plus d'informations sont disponibles. Ainsi dans [Aynaud, 2015], l'auteur propose une méthode pour calculer l'observabilité d'un amer (qui équivaut à une de nos parties) pour un capteur LIDAR en prenant en compte les différents éléments enregistrés dans la carte. Par exemple, un mur peut en cacher un autre selon l'angle de vue du véhicule considéré. Il est admis pour la suite qu'il sera toujours possible de calculer cette observabilité.

III.4.7.4.c Valeur de $\mathbf{d_{prec}}$

$\mathbf{d_{prec}}$ est le résultat de la détection précédente (qui a eu lieu à l'itération l_{prec}) si elle existe :

$$\mathbf{d_{prec}} = \begin{cases} false & \text{si la détection précédente a échoué} \\ true & \text{si la détection précédente a réussi} \end{cases}$$

Le résultat de la détection précédente $\mathbf{d_{prec}}$ n'a une influence que dans le cas où les conditions sont identiques, en conséquence l'information apportée par le triplet n'est pas nouvelle. S'il n'y a pas eu de détection précédente, l'information est nécessairement totalement nouvelle.

Nous avons donc présenté la construction de notre réseau bayésien pas à pas. Nous avons vu qu'il était possible de considérer de nouveaux événements en rajoutant des nœuds au réseau et en modifiant une partie des tables de probabilités conditionnelles

(seulement celles faisant intervenir le nouveau nœud). Nous avons également vu que supprimer des hypothèses (comme la non observabilité d'une partie lorsque l'estimation n'est pas intègre) pouvait se faire en modifiant une valeur dans une des tables. Avec l'ajout de nœuds et la modification des tables, il est possible d'obtenir une modélisation de plus en plus fine (et complexe) permettant de s'approcher de la réalité.

Ce réseau bayésien sert pour le critère de sélection pour calculer la probabilité de bonne détection (voir § III.3.9). Il sera également utilisé dans l'étape de mise à jour de notre algorithme. Nous allons d'ailleurs poursuivre le déroulement de notre algorithme avec l'étape de détection.

III.5 Détection

L'étape de sélection a eu lieu précédemment. Le meilleur triplet T_j a pu être sélectionné. En particulier, la partie à rechercher est déterminée ainsi que le capteur et le détecteur qui seront utilisés. Connaissant tout cela, il est possible de calculer une zone de recherche appelée également **zone de focalisation** dans laquelle le détecteur sera exécuté, c'est la *focalisation spatiale*. Le détecteur renverra un ensemble de primitives . Une *focalisation de caractéristiques* sera également faite à l'aide de l'état attendu pour la partie (voir le § II.3.2.2.d) soit avant la détection si le détecteur est paramétrable, soit après. Elle permettra de conserver uniquement les primitives compatibles avec la partie recherchée. S'il y a plusieurs primitives compatibles, l'une d'entre elles est choisie et la détection est réussie. S'il n'existe pas de primitives compatibles, la détection est un échec. Ce processus de détection avait été présenté rapidement dans le § II.3.3.4. La figure III.14 récapitule les différentes étapes de la détection que nous allons maintenant détailler.

III.5.1 Zone de focalisation

Comme nous l'avons déjà dit précédemment, le détecteur ne va pas être exécuté dans toute l'image capteur mais seulement dans une sous partie de celle-ci. Cela a deux avantages principaux : tout d'abord les temps de détection seront moindres puisqu'il ne sera pas nécessaire de traiter toute l'image ; ensuite cela augmente le rapport signal sur bruit ce qui permet d'améliorer le processus de détection. Cette focalisation est possible uniquement du fait du processus top-down guidé par l'objectif.

III.5.1.1 Exemple de l'intérêt de la focalisation spatiale

Nous allons illustrer l'intérêt de la focalisation spatiale sur une petite application de notre algorithme. Il s'agit de proposer un détecteur de visage vu de face (objet global) dans une image (la caméra est l'unique capteur) utilisant trois parties : une partie *faceComplète*, une partie *oeilGauche* et une partie *oeilDroit*. Nous disposons de deux détecteurs pour la partie *faceComplète* : le détecteur de face de OpenCv [Bradski, 2000] et le détecteur de face proposé dans la bibliothèque Dlib [4] [King, 2009]. Pour les parties *oeilGauche* et *oeilDroit*, le détecteur d'œil d'OpenCv est utilisé. Selon les critères, le détecteur de face d'OpenCv est

4. La bibliothèque Dlib, http://dlib.net/, est une bibliothèque générique, écrite en C++, facile à installer et à utiliser. Elle propose de nombreux outils. Elle offre notamment un outil graphique pour la création rapide de réseaux bayésiens et permet de calculer les probabilités associées à chaque nœud en prenant en compte les observations disponibles. Elle propose également un détecteur de visage vu de face déjà entraîné.

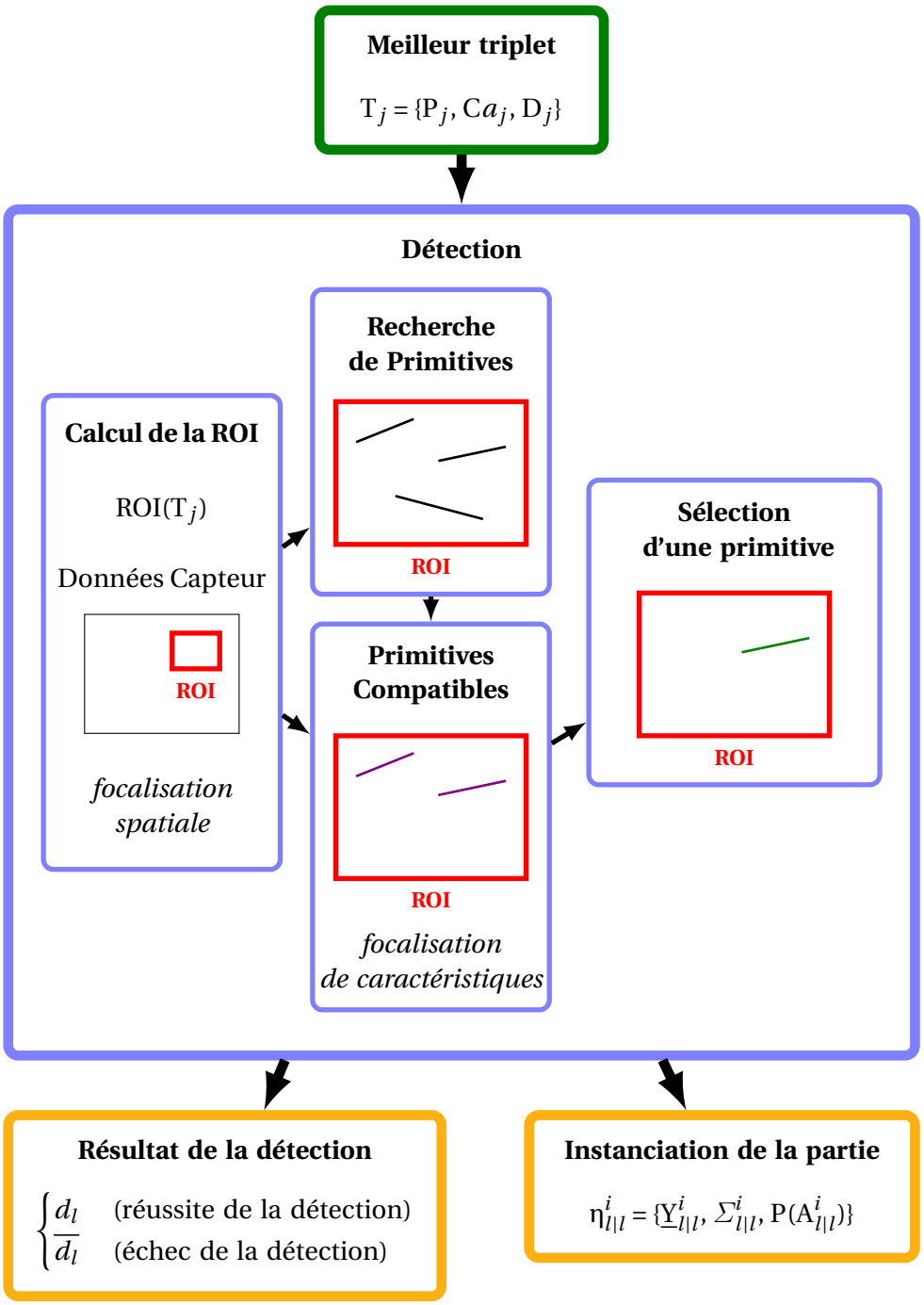


FIGURE III.14 – Schéma de principe de la détection : connaissant le triplet sélectionné, une région d'intérêt (ROI) est calculée, les primitives sont cherchées dans cette ROI. Les primitives compatibles sont extraites. Parmi les primitives compatibles, la meilleure est sélectionnée. Le résultat de la détection (échec ou réussite) est renvoyé. La partie recherchée est mise à jour.

appelé en premier. Les autres détecteurs seront donc uniquement appelés dans une zone restreinte.

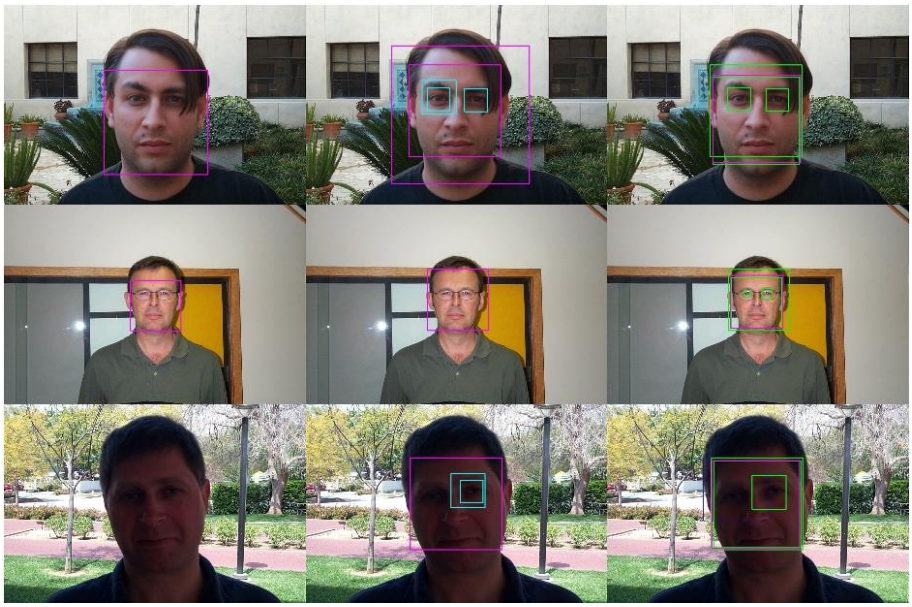


FIGURE III.15 – Illustration de l'intérêt de la focalisation spatiale. La première colonne représente les résultats renvoyés par le détecteur de face de la bibliothèque Dlib. La deuxième colonne montre les résultats des détecteurs d'OpenCv : en magenta la face et en cyan les yeux. La dernière colonne montre les résultats de notre algorithme : en magenta la face finale estimée, en vert les différentes parties trouvées, en bleu les parties non détectées.

La figure III.15 montre les résultats obtenus. La première ligne représente une situation où tous les détecteurs fonctionnent parfaitement. La deuxième ligne est un exemple où la focalisation spatiale améliore la détection : dans l'image entière (deuxième colonne), les yeux ne sont pas trouvés par le détecteur d'OpenCv alors que celui-ci parvient à les détecter lorsqu'il est utilisé dans une zone plus petite. Semblablement la troisième ligne montre un autre cas : le détecteur de face de la bibliothèque Dlib ne trouve rien dans toute l'image (première colonne) alors que là encore appelé dans une sous-image, il parvient à détecter.

III.5.1.2 Détermination de la zone de focalisation

Nous avons vu dans le § III.3.6.2 que la *focalisation spatiale* va découler de la *focalisation de caractéristiques*. Nous avons, lors de la *focalisation de caractéristiques* (voir le § III.3.6.1) , calculé un vecteur de caractéristiques attendu pour la partie avec sa covariance associée. La zone d'intérêt va donc être la zone minimale qui contient la *projection spatiale* de la partie moyenne ainsi que toutes ses variations admises décrites grâce à la matrice de covariance associée.

La figure III.16 montre un exemple : les variations autorisées pour chaque coin du carré à trouver sont représentées par des ellipses oranges. La région d'intérêt (appelée également **zone de focalisation spatiale**) pour chaque coin est donc l'ellipse orange associée.

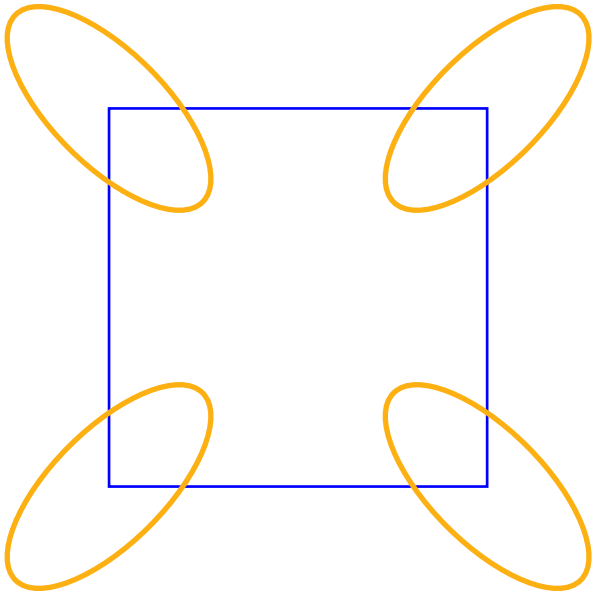
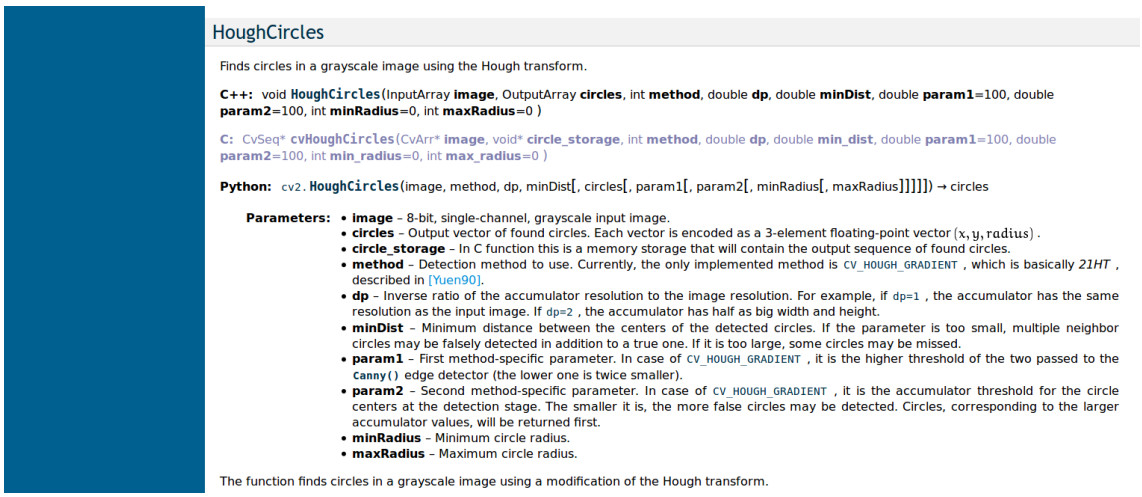


FIGURE III.16 – Zones de focalisation pour les 4 coins d'un carré. Le carré "moyen" est dessinée en bleu. Les zones de focalisation associées à chacun de ses coins sont représentées par les ellipses oranges.

Il faut noter que si par exemple, les coins du carré étaient de couleur rouge, cela ne modifierait pas la zone de focalisation spatiale alors que la focalisation de caractéristiques serait affectée. Seules certaines caractéristiques (celles qui influent sur la position ou sur la taille de la zone de recherche de la partie) seront prises en compte : c'est la *projection spatiale*.

III.5.1.3 Spécialisation des détecteurs

Il est parfois possible d'avoir des détecteurs spécialisables qui pourront profiter de la *focalisation de caractéristiques* (voir le §II.3.3.4.b et le § III.3.6.1) mentionnée précédemment.



HoughCircles

Finds circles in a grayscale image using the Hough transform.

C++: void **HoughCircles**(InputArray **image**, OutputArray **circles**, int **method**, double **dp**, double **minDist**, double **param1**=100, double **param2**=100, int **minRadius**=0, int **maxRadius**=0)

C: CvSeq* **cvHoughCircles**(CvArr* **image**, void* **circle_storage**, int **method**, double **dp**, double **min_dist**, double **param1**=100, double **param2**=100, int **min_radius**=0, int **max_radius**=0)

Python: cv2.**HoughCircles**(image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]]) → circles

Parameters:
- **image** – 8-bit, single-channel, grayscale input image.
- **circles** – Output vector of found circles. Each vector is encoded as a 3-element floating-point vector $(x, y, radius)$.
- **circle_storage** – In C function this is a memory storage that will contain the output sequence of found circles.
- **method** – Detection method to use. Currently, the only implemented method is `CV_HOUGH_GRADIENT` , which is basically *21HT* , described in [Yuen90].
- **dp** – Inverse ratio of the accumulator resolution to the image resolution. For example, if `dp=1` , the accumulator has the same resolution as the input image. If `dp=2` , the accumulator has half as big width and height.
- **minDist** – Minimum distance between the centers of the detected circles. If the parameter is too small, multiple neighbor circles may be falsely detected in addition to a true one. If it is too large, some circles may be missed.
- **param1** – First method-specific parameter. In case of `CV_HOUGH_GRADIENT` , it is the higher threshold of the two passed to the `Canny()` edge detector (the lower one is twice smaller).
- **param2** – Second method-specific parameter. In case of `CV_HOUGH_GRADIENT` , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.
- **minRadius** – Minimum circle radius.
- **maxRadius** – Maximum circle radius.

The function finds circles in a grayscale image using a modification of the Hough transform.

FIGURE III.17 – Copie de la documentation d'OpenCv sur son détecteur de cercle

Par exemple, si nous regardons la documentation d'OpenCv [Bradski, 2000] (voir figure III.17) pour son détecteur de cercles, nous voyons une liste de paramètres parmi lesquels $minRadius$ et $maxRadius$. Ces deux paramètres correspondent respectivement au rayon minimal et maximal pour les cercles recherchés. Il est donc possible de spécialiser ce détecteur de cercles lorsqu'il y a des informations sur le rayon attendu. Par exemple, si un objet "carré" est cherché, avec un cercle inscrit à l'intérieur de rayon le côté du carré, une fois un bord du carré trouvé, le rayon du cercle est pratiquement connu (il peut y avoir une erreur de quelques pixels liée à la précision des différents détecteurs) : il est donc possible d'attribuer une valeur au rayon minimal et maximal souhaités.

Nous allons maintenant définir ce qu'est une primitive compatible.

III.5.2 Primitives compatibles

Une primitive compatible est une primitive dont les caractéristiques correspondent aux caractéristiques attendues pour la partie cherchée. Par exemple, pour une partie *segmentHorizontal*, parmi l'ensemble des segments, seuls ceux qui auront une pente proche de 0 pourront être des primitives compatibles.

TABLEAU III.16 – Notations pour les primitives compatibles

$d_{\mathrm{P}rim}$	distance de Mahalanobis entre le vecteur de caractéristiques de la primitive $\mathrm{P}rim$ renvoyée par le détecteur et celui attendu pour la partie i.
$\mathrm{P}rim$	une primitive renvoyée par le détecteur
$\underline{\mathrm{Y}}_{\mathrm{P}rim}$	vecteur de caractéristiques de la primitive $\mathrm{P}rim$
$\mathbf{M}_{aug}$	matrice de covariance représentant les variations tolérées.
$\mathbf{M}_{\mathrm{D}_j}$	matrice représentant la précision du détecteur D_j avec lequel la détection est réalisée
S	seuil d'acceptabilité pour considérer une primitive compatible
$\mathrm{P}rim_{best}$	meilleure primitive compatible au sens de la distance de Mahalanobis

La définition de primitive compatible est importante pour tous les détecteurs. En effet, soit les détecteurs sont paramétrables et dans ce cas, la *focalisation de caractéristiques* sera utile pour les spécialiser. Les primitives renvoyées par ces détecteurs seront nécessairement compatibles avec la partie. Soit les détecteurs ne sont pas paramétrables et dans ce cas, il faudra trier les primitives pour ne garder que les primitives compatibles avec la partie. C'est par exemple la cas pour un détecteur de segments qui renverrait tous les segments présents dans une zone rectangulaire.

Pour déterminer si une primitive est compatible, une distance de Mahalanobis entre le vecteur de caractéristiques attendu pour la partie $\underline{\mathrm{Y}}^i_{l|l-1}$ (voir le § III.3.6.1) et le vecteur de caractéristiques de la primitive $\underline{\mathrm{Y}}_{\mathrm{P}rim}$ est calculée :

$$d_{\mathrm{P}rim} = \sqrt{(\underline{\mathrm{Y}}_{\mathrm{P}rim} - \underline{\mathrm{Y}}^i_{l|l-1})^{\mathrm{T}} \mathbf{M}^{-1}_{aug} (\underline{\mathrm{Y}}_{\mathrm{P}rim} - \underline{\mathrm{Y}}^i_{l|l-1})} \tag{III.10}$$

où

— $d_{\mathrm{P}rim}$ est la distance de Mahalanobis pour la primitive considérée

— $\mathbf{M}_{aug}$ est la matrice de covariance représentant les variations tolérées.

$$\mathbf{M}_{aug} = \mathbf{M}_{\mathrm{D}_j} + \Sigma^i_{l|l-1} \tag{III.11}$$

où

— $\mathbf{M}_{\mathrm{D}_j}$ est la matrice représentant la précision du détecteur D_j (avec lequel la détection est réalisée) pour chaque paramètre

— $\Sigma^i_{l|l-1}$ est la matrice de covariance représentant les variations tolérées pour la partie i=P_j

— $\underline{\mathrm{Y}}^i_{l|l-1}$ est le vecteur de caractéristique attendu pour la partie i=P_j

Une primitive sera considérée compatible uniquement si :

$$d_{\mathrm{P}rim} \leq \mathrm{S}$$

où S est le seuil d'acceptabilité (fixé).

La valeur du seuil va donc influencer la détection en permettant d'accepter ou de rejeter les différentes primitives. Un seuil trop bas conduira à rejeter des primitives qui correspondaient réellement à la partie cherchée et donc nuira à la détection de l'objet global en empêchant la détection de certaines de ses parties. À l'inverse, un seuil trop élevé fera perdre du temps en tolérant des primitives erronées qui seront alors choisies et conduiront à de mauvaises solutions avant d'être remises en cause. À l'issue de tests expérimentaux, la valeur S est fixée à 3.

Il est possible qu'un détecteur fournisse plusieurs primitives candidates. Dans ce cas, la meilleure au vu de la relation (III.10) sera choisie.

La détection est considérée comme réussie si au moins une primitive compatible a été trouvée. Il existe au moins une primitive $\mathrm{P}rim$ telle que $d_{\mathrm{P}rim} \leq \mathrm{S}$.

III.5.3 Mise à jour de l'état de la partie après détection

En fonction de l'échec ou de la réussite de la détection, la partie i sera mise à jour. L'état de la partie i, après cette mise à jour, sera noté $\eta^i_{l|l}$. Nous allons étudier les deux cas possibles pour la mise à jour : le premier cas est celui où la détection a échoué et le second cas est celui où la détection a réussi.

III.5.3.1 Cas de l'échec de la détection

Aucune primitive compatible pour la partie n'a été trouvée. Un indicateur (booléen) sera renvoyé pour signaler l'échec de la détection. Le nouvel état de la partie sera :

$$\eta^i_{l|l} = \{\underline{\mathrm{Y}}^i_{l|l}, \Sigma^i_{l|l}, \mathrm{P}(\mathrm{A}^i_{l|l}))\}$$

avec :

— $\underline{\mathrm{Y}}^i_{l|l} = \underline{\mathrm{Y}}^i_{l|l-1}$: la valeur prédite est conservée à défaut d'avoir plus de données.

— $\Sigma^i_{l|l} = \Sigma^i_{l|l-1}$: idem.

— $P(A^i_{l|l}) = P(A^i_{l|l-1}|\overline{d_l})$: le réseau bayésien (voir figure III.18) est utilisé pour calculer la nouvelle valeur de la probabilité d'existence de la partie sachant l'échec de la détection.

Il est possible de calculer la probabilité a posteriori, $P(A^i_{l|l})$, que la partie i existe en utilisant le principe de l'**inférence bayésienne** avec l'**évidence** [5] liée au résultat de la détection. En effet, on **sait** désormais si la détection a réussi ou non, ce n'est plus un événement incertain. Il a été observé.

III.5.3.2 Cas de la réussite de la détection

La primitive $Prim_{best}$ sélectionnée précédemment est utilisée pour réaliser la mise à jour de la partie. Le nouvel état de la partie sera :

$$\eta^i_{l|l} = \{\underline{Y}^i_{l|l}, \Sigma^i_{l|l}, P(A^i_{l|l})\}$$

— $\underline{Y}^i_{l|l} = \underline{Y}_{Prim_{best}}$: la valeur du vecteur caractéristiques $\underline{Y}_{Prim_{best}}$ de la primitive $Prim_{best}$ est affectée à la partie i.

— $\Sigma^i_{l|l} = \mathbf{M}_{D_j}$: la matrice liée à la précision du détecteur est affectée à la matrice de covariance de la partie i.

— $P(A^i_{l|l}) = P(A^i_{l|l-1}|d_l)$: le réseau bayésien (voir figure III.18) est utilisé pour calculer la nouvelle valeur de la probabilité d'existence de la partie sachant, cette fois-ci, que la détection a réussi.

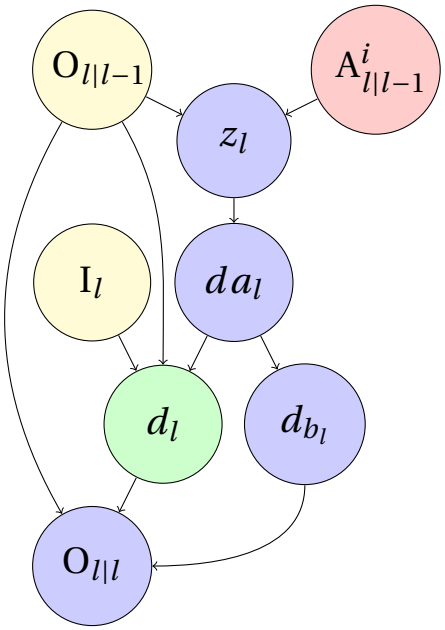


FIGURE III.18 – Réseau bayésien permettant de connaître a posteriori la probabilité d'existence de la partie $P(A^i_{l|l})$ (nœud en rose) sachant le résultat d_l de la détection (nœud en vert). En jaune, les nœuds d'entrée pour lesquels la probabilité est donnée au réseau bayésien. La probabilité $P(O_{l|l-1})$ correspond à la confiance dans l'état prédit pour l'itération l. $P(I_l)$ représente la probabilité que l'information apportée par le triplet soit nouvelle (voir § III.3.7).

Dans les deux cas, l'état de la partie i peut être mis à jour.

Une fois ce processus de détection terminé, nous allons pouvoir effectuer l'étape de mise à jour de l'état global.

5. Avoir une évidence pour un événement signifie connaître sa réalisation. Pour un événement binaire, il s'agit d'être capable de savoir s'il est vrai ou faux.

III.6 Mise à jour de l'état global

La mise à jour se décompose en deux mises à jour distinctes, comme cela a déjà été dit dans le le chapitre précédent dans le § II.3.3.5. La première concerne la mise à jour de la confiance tandis que la seconde est une mise à jour de l'estimation (voir figure III.19). Nous allons donc maintenant étudier plus en détails ces mises à jour.

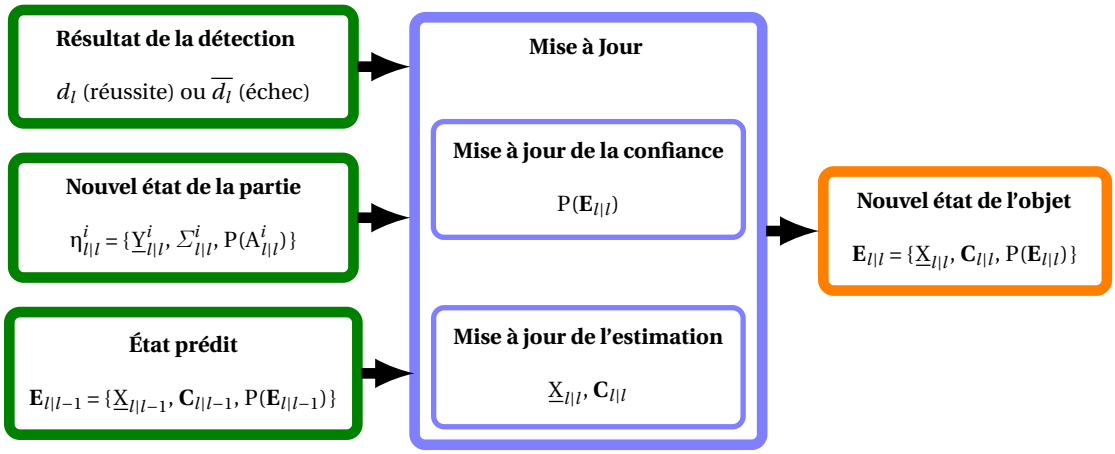


FIGURE III.19 – Schéma de principe de la mise à jour

III.6.1 Mise à jour de la confiance

Nous avons rapidement évoqué dans le § II.3.3.5 que différents éléments allaient avoir une influence sur l'évolution de la confiance. Nous allons les présenter succinctement ici.

- ***Confiance dans l'état prédit***
  Il est plus probable de réussir une détection si l'estimation, avant mise à jour, est intègre (événement $O_{l|l-1}$). En conséquence, la confiance dans l'intégrité de l'estimation sera a posteriori plus élevée que la confiance a priori en cas de réussite de la détection : $P(O_{l|l-1}|d) \geq P(O_{l|l-1})$.

- ***Nombre de candidats possibles***
  En fonction du nombre de primitives compatibles renvoyées par le détecteur, la confiance $P(\mathbf{E}_{l|l})$ dans la nouvelle estimation sera plus faible que la confiance $P(\mathbf{E}_{l|l-1})$ dans l'estimation avant mise à jour (et ce donc malgré la réussite de la détection). En effet, puisqu'il a plusieurs primitives candidates, rien ne garantit que le "bon" choix ait été fait.

- ***Qualité du détecteur***
  En fonction de la qualité du détecteur, caractérisée par les paramètres α et β (voir le § III.4.3), le fait de réussir ou non une détection aura plus ou moins d'influence sur l'évolution de la confiance. Par exemple, si α est élevé alors il y a de fortes chances de ne pas voir la primitive même si elle était présente. Ne rien trouver ne sera donc pas très pénalisant pour la confiance.

- ***Observabilité de la partie recherchée***
  Ne pas voir une partie difficilement observable, par exemple parce qu'elle est probablement occultée, aura moins d'influence sur la confiance que de ne pas voir une partie parfaitement observable.

- ***Existence de la partie recherchée***
  Ne pas voir une partie optionnelle (comme par exemple une *paire de lunettes* pour un visage) sera moins grave que ne pas voir une partie dont l'existence est certaine (comme une *bouche* pour un visage). La confiance diminuera donc moins en cas d'échec de la détection dans le premier cas que dans le second.

- ***Nouveauté de l'information***
  Comme nous l'avons dit précédemment (voir le § III.3.7), s'il y a déjà eu une tentative de détection pour le triplet à l'itération l_{prec} et que les conditions de détection à l'itération l sont exactement les mêmes alors le résultat de la nouvelle détection devrait être exactement le même et la confiance ne devrait pas changer.

Ces différents événements sont justement ceux pris en compte par le réseau bayésien construit dans la section III.4. Pour mettre à jour la confiance, nous allons donc utiliser notre réseau bayésien. Le graphe du réseau est à nouveau présenté ici en mettant en exergue les éléments importants pour la mise à jour de la confiance.

Comme nous pouvons le voir dans la figure III.20, la probabilité qui nous intéresse et qui va nous permettre de calculer la nouvelle confiance est la probabilité du nœud $O_{l|l}$. La nouvelle confiance $P(\mathbf{E}_{l|l})$ est alors simplement :

$$\begin{aligned} P(\mathbf{E}_{l|l}) &= P(O_{l|l}|\overline{d_l}) \quad \text{si la détection a échoué} \\ P(\mathbf{E}_{l|l}) &= P(O_{l|l}|d_l) \quad \text{si la détection a réussi} \end{aligned} \tag{III.12}$$

La structure du réseau reste la même dans les deux cas.

III.6.2 Mise à jour de l'estimation

Pour la mise à jour de l'estimation, là encore, le processus va dépendre de la réussite ou non de la détection.

III.6.2.0.a Cas de l'échec de la détection

Encore une fois, l'échec de la détection est le cas le plus simple. Si la détection a échoué, aucune primitive compatible n'a pu être trouvée. Il n'y a donc aucune nouvelle donnée nous permettant d'affiner l'estimation. En conséquence :

$$\begin{aligned} \underline{X}_{l|l} &= \underline{X}_{l|l-1} \\ \mathbf{C}_{l|l} &= \mathbf{C}_{l|l-1} \end{aligned} \tag{III.13}$$

III.6.2.0.b Cas de la réussite de la détection

Pour faire la mise à jour de l'estimation, nous avons choisi d'utiliser un filtre de Kalman présenté dans l'annexe B.

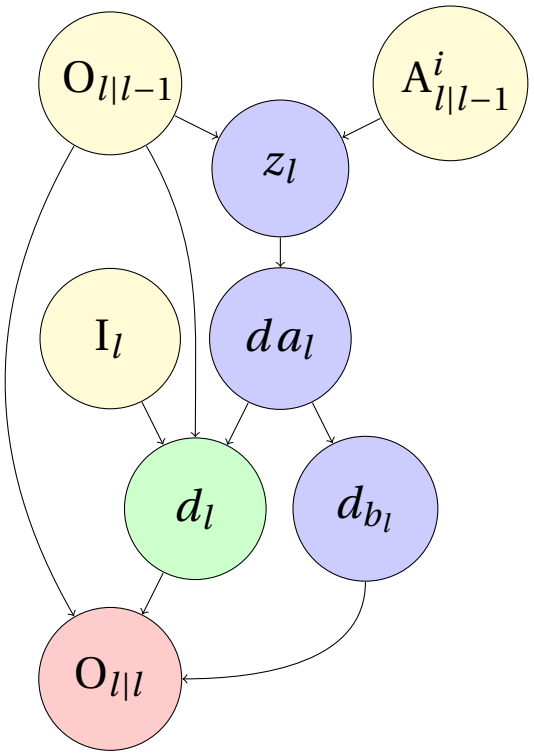


FIGURE III.20 – Réseau bayésien permettant de connaître a posteriori que l'estimation après mise à jour soit intègre $P(O_{l|l})$ (nœud en rose) sachant le résultat d_l de la détection (nœud en vert). En jaune, les nœuds d'entrée pour lesquels la probabilité est donnée au réseau bayésien. La probabilité $P(O_{l|l-1})$ correspond à la confiance dans l'état prédit pour l'itération l. $P(I_l)$ représente la probabilité que l'information apportée par le triplet soit nouvelle. Enfin, la probabilité $P(A^i_{l|l-1})$ correspond à la probabilité que la partie cherchée existe.

Nous avons besoin des fonctions $\mathbf{h}_i$ et des jacobiennes associées $\mathbf{H}_i$ (voir le § II.3.2.2) qui permettent de relier le vecteur de caractéristiques de l'objet $\underline{X}$ au vecteur de caractéristiques de la partie $\underline{Y}$. L'état du système pour le filtre de Kalman correspond donc au vecteur de caractéristiques de l'objet. L'observation sera le vecteur de caractéristiques de la partie après la mise à jour liée à la détection d'une primitive : $\underline{Y}^i_{l|l}$. Pour le bruit, la matrice de covariance associée $\Sigma^i_{l|l}$, qui correspond également à l'erreur du détecteur sur la mesure (voir le § III.5.3) sera utilisée.

Comme mentionné dans le § II.3.2.1.f , nous admettons disposer d'une fonction d'évolution **f** et de sa jacobienne associée **F** qui sont nécessaires pour la phase de prédiction du filtre de Kalman. Ainsi, il est possible de faire la mise à jour de l'estimation.

III.6.3 Remise en cause

Il est possible que l'algorithme fasse une erreur d'association à un moment donné : en effet lorsqu'il y a plusieurs primitives acceptables pour une partie, rien ne garantit que ce soit la **bonne** primitive qui ait été sélectionnée.

Or lorsque le choix est mauvais, l'estimation a de grandes chances de ne plus être intègre. La suite des détections est compromise. Les zones de focalisation ne sont plus valides. Elles ne sont plus centrées autour de la primitive réellement cherchée. Les détections ultérieures ont alors de grandes chances d'échouer. Afin d'éviter d'être dans une impasse à cause d'une mauvaise association, nous proposons un mécanisme de retour qui va permettre de modifier les associations faites dans le passé.

III.6.3.1 Conditions de Retour

Ce processus de retour sera utile si premièrement la détection a échoué et s'il était plus probable que cette non détection provienne de la non intégrité de l'estimation (supposée liée à une mauvaise association lors d'une précédente itération) plutôt que d'un autre événement (non existence de la partie ou erreur du détecteur par exemple).

Pour cela, nous utilisons le réseau bayésien construit dans le § III.4 afin de calculer les différentes probabilités après inférence bayésienne (on **sait** que la détection a échoué) et ainsi déterminer la cause la plus probable de la non détection. La figure III.21 montre les nœuds qui vont servir à déterminer s'il faut effectuer un processus de retour.

Le retour en arrière se fera si $r_{Retour} = true$ avec :

$$\begin{aligned} r_{Retour} \;=\; & \mathrm{P}(\overline{\mathrm{A}^i_{l|l-1}}|\overline{d_l}) && \leq \mathrm{P}(\overline{\mathrm{O}_{l|l-1}}|\overline{d_l}) \\ & \text{ET } \alpha && \leq \mathrm{P}(\overline{\mathrm{O}_{l|l-1}}|\overline{d_l}) \\ & \text{ET } \mathrm{P}(\overline{z_l}|\overline{d_l}) && \leq \mathrm{P}(\overline{\mathrm{O}_{l|l-1}}|\overline{d_l}) \end{aligned} \tag{III.14}$$

La probabilité $\mathrm{P}(\mathrm{A}^i_{l|l-1}|\overline{d_l})$ que l'estimation n'était pas intègre sachant que la détection a échoué doit donc être supérieure à la probabilité α que le détecteur n'ait pas vu la primitive alors qu'elle était présente, à la probabilité $\mathrm{P}(\overline{\mathrm{A}^i_{l|l-1}}|\overline{d_l})$ que la partie cherchée n'existait pas et à la probabilité $\mathrm{P}(\overline{z_l}|\overline{d_l})$ que la partie n'était pas observable dans la modalité capteur sélectionnée. Donc si la cause la plus probable de la non détection était la non intégrité de l'estimation.

r_{Retour} est un booléen indiquant qu'il est nécessaire de revenir en arrière.

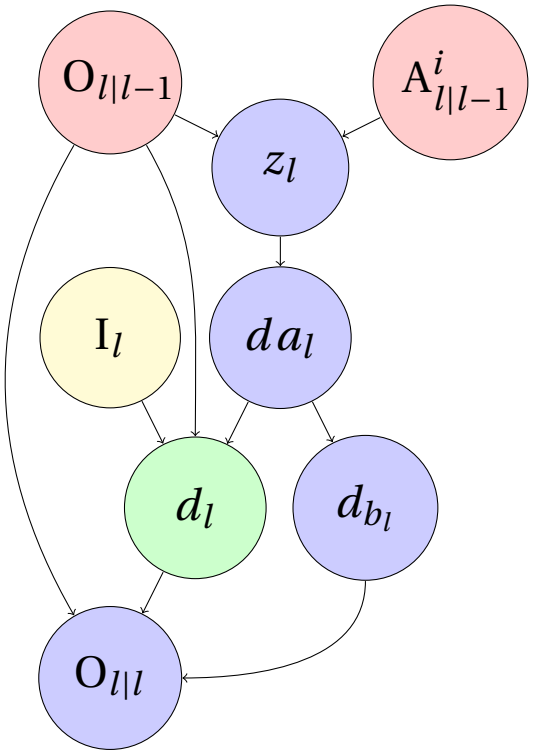


FIGURE III.21 – Réseau bayésien permettant de déterminer si un retour en arrière est nécessaire. Nous nous intéressons, ici, à la probabilité a posteriori d'existence de la partie (nœud $A^i_{l|l-1}$) et à la probabilité a posteriori que l'estimation était intègre (nœud $O_{l|l-1}$), sachant le résultat d_l de la détection (nœud en vert). En jaune, le nœud d'entrée pour lequel la probabilité est donnée au réseau bayésien. $P(I_l)$ représente la probabilité que l'information apportée par le triplet soit nouvelle.

III.6.3.2 Indice de retour

Maintenant qu'il est possible de savoir si le processus de retour est nécessaire, il faut déterminer à quelle étape précédente revenir. Par exemple, si à l'étape précédente, il y avait aussi eu une non détection (mais le processus de retour n'avait pas été déclenché), revenir à cette étape ne nous apportera rien. Nous cherchons à corriger les problèmes liés à de mauvaises associations. Pour cela, le processus de retour devra permettre de revenir à l'itération r telle que

— la détection avait réussi

— plusieurs primitives étaient compatibles avec la partie cherchée

Ce processus peut remonter jusqu'à l'initialisation. Toutefois, l'utilisateur peut, en fonction de l'application visée et de la mémoire de traitement disponible, définir un horizon temporel au delà duquel il ne peut remonter.

III.6.3.3 Mise à jour de retour

Évidemment, le retour ne doit pas juste consister à revenir exactement dans le même état que celui de l'indice de retour. Il faut modifier la confiance pour prendre en compte les échecs de détection. Il s'agit de mettre à jour l'ensemble des probabilités passées prenant en compte les diverses observations réalisées (avec notamment les non détections observées).

Pour faire cela, il faudrait idéalement chaîner l'ensemble des réseaux bayésiens de chaque étape l comme montré dans la figure III.22. Le réseau à l'étape l, noté R_l pour la suite, est utilisé pour mettre à jour le réseau R_{l-1} qui lui-même permettra de mettre à jour le réseau R_{l-2} etc.

Pour faire ce chaînage, il faut donc lier les réseaux successifs entre eux. La table III.17 de probabilités conditionnelles définit les liens entre deux réseaux successifs R_{g-1} et R_g avec $g \in [2, l]$. Pour rappel, $\mathrm{O}_{g-1|g-1}$ désigne l'événement : *l'estimation définie par* $\underline{\mathrm{X}}_{g-1|g-1}$ *et* $\boldsymbol{C}_{g-1|g-1}$ *est intègre* tandis que l'événement $\mathrm{O}_{g|g-1}$ désigne l'événement : *l'estimation définie par* $\underline{\mathrm{X}}_{g|g-1}$ *et* $\boldsymbol{C}_{g|g-1}$ *est intègre.*

Nous faisons l'hypothèse que le processus de prédiction est cohérent et conserve l'intégrité, c'est à dire que si $\mathrm{O}_{g-1|g-1}$ était vrai alors $\mathrm{O}_{g|g-1}$ l'est aussi (ce qui correspond à la deuxième ligne de la table III.17). Si l'estimation n'était pas intègre ($\mathrm{O}_{g-1|g-1} = 0$), alors elle ne le deviendra pas avec la prédiction ($\mathrm{O}_{g|g-1} = 0$), c'est ce qui permet d'écrire la première ligne de la table III.17.

TABLEAU III.17 – Table de probabilités conditionnelles pour le chaînage des réseaux R_{g-1} et R_g avec $g \in [2, l]$.

$\mathrm{O}_{g-1\|g-1}$	$\mathrm{O}_{g\|g-1}$
0	0
1	1

Dans l'hypothèse où nous souhaiterions effectuer globalement une inférence de tous les nœuds du réseau global (formé de l'ensemble des réseaux R_1 à R_l), les calculs engendrés seraient beaucoup trop importants. Nous allons mettre à jour de manière rétro active le réseau à l'étape $l-1$ à l'aide de celui de l'étape l et ainsi de suite jusqu'à l'horizon souhaité.

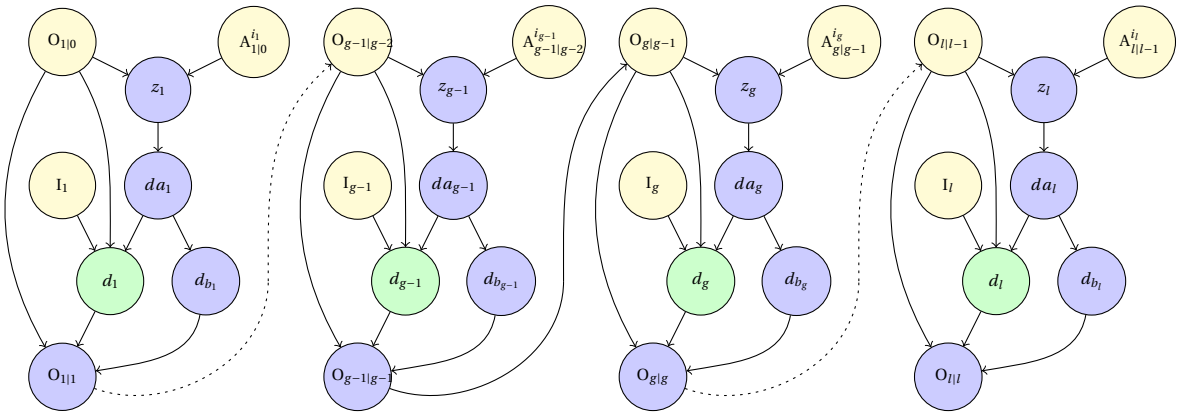


FIGURE III.22 – Chaînage des réseaux bayésiens R_1 à R_l

La complexité de ce processus de retour sera alors linéaire (en fonction de la taille de l'horizon) plutôt que combinatoire.

Pour ce faire, nous créons une évidence *virtuelle* [Pearl, 1988]. L'idée est la suivante, l'enchaînement de deux réseaux tels que présentés dans la figure III.23 est remplacé par un unique réseau comme celui de la figure III.24.

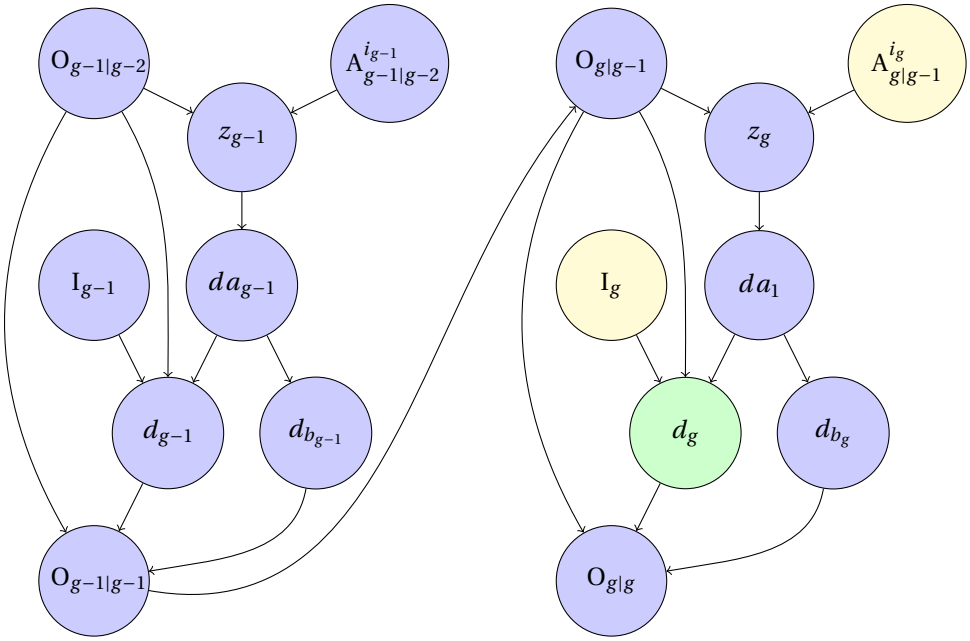


FIGURE III.23 – chaînage des réseaux bayésiens R_{g-1} et R_g pour $g \in [2, l]$

Le principe est le suivant : une observation a été réalisée à l'étape g mais nous voudrions connaître l'influence de cette observation sur le nœud $O_{g-1|g-1}$ (correspondant donc à l'étape $g-1$). Le lien entre les réseaux R_{g-1} et R_g est donné par la table III.17 et nous donne que $P(O_{l-1|l-1}) = P(O_{l|l-1})$. Donc nous aurons également $P(O_{l-1|l-1}|\overline{d_{g+1}}) = P(O_{l|l-1}|\overline{d_{g+1}})$. La probabilité $P(O_{l|l-1}|\overline{d_{g+1}})$ peut être calculée avec notre réseau bayésien classique (celui de la figure III.21) grâce au procédé d'inférence bayésienne. Nous connaissons donc la valeur de la probabilité $P(O_{l-1|l-1}|\overline{d_{g+1}})$, appelons la p_{voulue}. Nous voulons dans le réseau de

la figure III.24 utiliser le nœud V comme une évidence afin d'avoir $P(O_{l-1|l-1}|V) = p_{voulue}$.

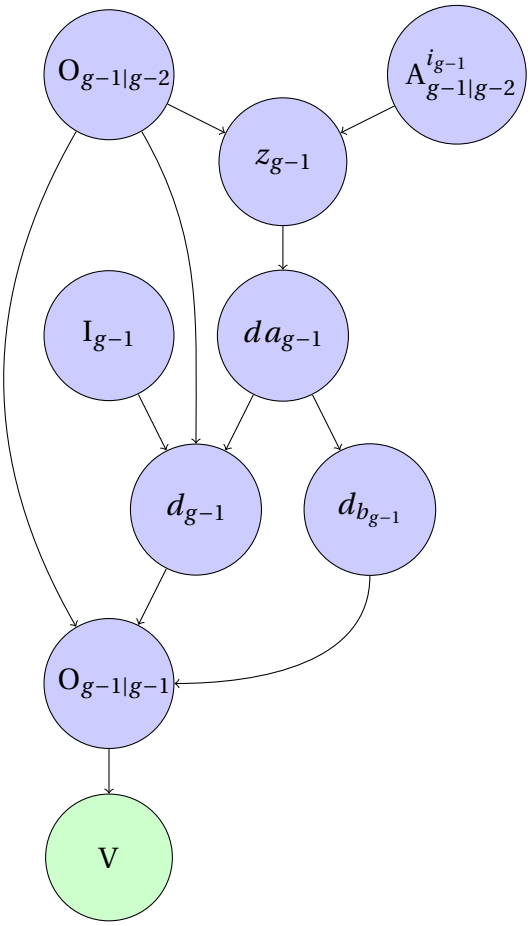
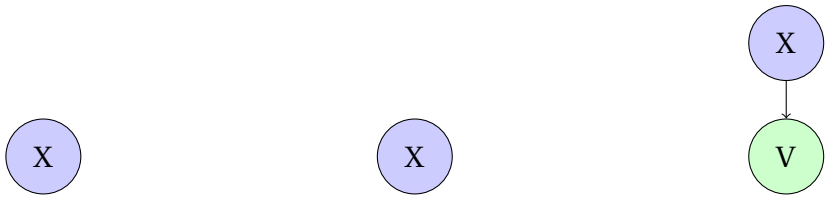


FIGURE III.24 – Réseau bayésien R_{g-1} avec un noeud V utilisé comme évidence virtuelle.

Pour faciliter la compréhension, nous allons détailler avec un réseau plus simple ce que nous désirons faire. Nous désirons fixer la probabilité d'un nœud X représentant un événement binaire à une valeur différente de 0 ou 1. Pour cela, nous définissons un nouveau nœud V virtuel configuré de telle sorte qu'une évidence sur ce nœud force la probabilité du nœud X à la valeur voulue. La figure III.25 illustre ce procédé. Dans la sous figure III.25a, le nœud tel qu'il est au départ est représenté. La probabilité que l'événement X soit vrai est de p_x. L'objectif est d'obtenir pour la probabilité que l'événement X soit vrai une valeur p_{voulue}. Pour cela, un nœud V virtuel (dans le sens où il ne correspond à aucun événement) est rajouté de manière à contraindre la probabilité de X (voir la sous figure III.25c). Ce nœud va être fixé comme étant une inférence bayésienne, d'où le nom d'évidence virtuelle adopté dans [Pearl, 1988].

Afin d'avoir le comportement souhaité, il faut donc définir les probabilités conditionnelles de l'événement V. Admettons avoir $p_{voulue} \neq 0$ et $p_{voulue} \neq 1$. Soit γ tel que : $0 < \gamma < \min(\frac{p_x}{p_{voulue}}, \frac{1-p_x}{1-p_{voulue}})$. Alors, la table III.18 de probabilité conditionnelle définie donne le résultat escompté (nous aurons $P(X|V) = p_{voulue}$).



(a) Réseau de base à un nœud (b) Réseau souhaité (c) Ajout d'une évidence virtuelle V

FIGURE III.25 – Utilisation d'une évidence virtuelle. La sous figure III.25a montre le réseau originel. La sous figure III.25b montre le réseau que nous souhaiterions avoir. La sous figure III.25c montre l'ajout d'un nœud d'évidence virtuelle afin d'obtenir le réseau souhaité.

TABLEAU III.18 – Table de probabilités conditionnelles pour le réseau bayésien de la sous figure III.25c.

X	V
0	$c = \gamma \frac{1-p_{voulue}}{1-p_x}$
1	$d = \gamma \frac{p_{voulue}}{p_x}$

avec $\gamma \in]0, \min(\frac{p_x}{p_{voulue}}, \frac{1-p_x}{1-p_{voulue}})[$

III.6.3.3.a Démonstration

De manière évidente, avec la définition de p_x, p_{voulue} et les conditions sur γ, $c = \gamma \frac{1-p_{voulue}}{1-p_x}$ et $d = \gamma \frac{p_{voulue}}{p_x}$ sont bien compris entre 0 et 1. D'après la formule des probabilités totales (voir le § C.2.3.1 dans l'annexe C) :

$$\begin{aligned} P(V) &= P(V|\overline{X})P(\overline{X}) + P(V|X)P(X) \\ &= c\,(1-p_x) + d\,p_x \\ &= \gamma \frac{1-p_{voulue}}{1-p_x}(1-p_x) + \gamma \frac{p_{voulue}}{p_x} p_x \\ &= \gamma \end{aligned}$$

D'après le théorème de Bayes (voir le § C.2.3.2 dans l'annexe C) :

$$\begin{aligned} P(X|V) &= \frac{P(V|X)P(X)}{P(V)} \\ &= \frac{d\,p_x}{P(V)} \\ &= \gamma \frac{p_{voulue}}{p_x} \frac{p_x}{\gamma} \\ &= p_{voulue} \end{aligned}$$

C'est bien le résultat souhaité. Il suffit donc de choisir un γ respectant les conditions posées pour résoudre le problème. Le même procédé peut être appliqué au chaînage des deux réseaux successifs de la figure III.23 et de manière plus générale, en le répétant autant de fois que nécessaire, pour plusieurs réseaux successifs. Pour obtenir la confiance à l'étape de retour, notée r, il suffit donc de procéder itérativement jusqu'à parvenir au réseau R_r.

III.6.3.3.b Quelques remarques

Nous avons montré comment calculer la nouvelle confiance après retour. Néanmoins, il reste quelques autres changements à effectuer. En effet, par le biais du retour, les actions faites entre l'étape de retour r et l'étape actuelle l sont partiellement annulées. En particulier, les probabilités des parties pour lesquelles une détection avait eu lieu seront remises à la valeur qu'elles avaient à l'étape r.

De plus, la primitive qui avait été initialement choisie à l'étape r va être marquée afin de ne plus la sélectionner à nouveau. Il restera donc uniquement ces voisines comme choix possibles. Il faut remarquer que l'algorithme déroulera à nouveau le processus de sélection, ainsi même si c'était le triplet T_r qui auparavant avait été choisi, il sera possible qu'un autre triplet T_j soit finalement pris. Cela pourra être le cas, par exemple, lorsque le nombre de voisins **nbVoisins** avait mal été estimé et se révèle finalement beaucoup plus élevé que prévu, diminuant ainsi l'attrait de T_r.

III.7 Critère de Fin

Une fois la mise à jour de l'état global achevé, il est nécessaire de déterminer si l'algorithme doit ou non se terminer. Pour cela, un critère de fin est utilisé. Comme pour les blocs précédents, un schéma succinct est proposé dans la figure III.26

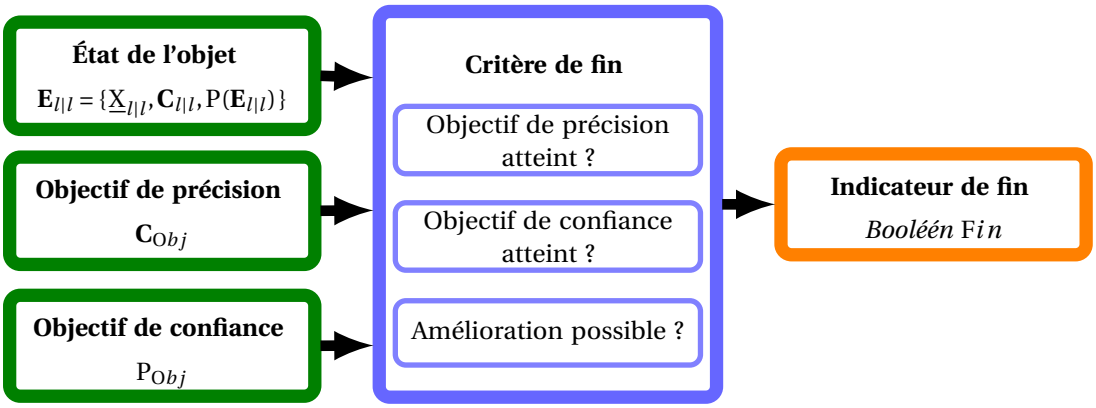


FIGURE III.26 – Schéma de principe pour le critère de fin

Naturellement, l'algorithme va se terminer lorsque l'objectif de précision **et** l'objectif de confiance seront atteints (voir le § III.3.2). L'un et l'autre sont complémentaires. Savoir que le véhicule est sur la Terre est sans grand intérêt d'un point de vue localisation, pourtant, la confiance associée sera maximale (cette estimation est absolument certaine). À l'inverse, avoir une précision de 0.1 cm est inutile s'il y a peu de chances que cela corresponde à la réalité. Les deux sont donc bien nécessaires pour décider de la terminaison de l'algorithme. Tant que l'un ou l'autre ne sera pas satisfait, de nouvelles détections devront être tentées : soit afin d'améliorer la précision, soit afin d'augmenter la confiance. Chaque détection réussie doit influencer positivement au moins l'une des deux composantes. L'algorithme se terminera donc lorsque :

$$P(\mathbf{E}_{l|l}) \geq P_{Obj}$$
$$\text{ET} \quad \mathbf{C}_{l|l}(m,m) \leq \mathbf{C}_{Obj}(m,m), \forall m \in [1,n]$$

Néanmoins, il peut arriver que les objectifs ne puissent pas être atteints. En effet, ils peuvent être trop ambitieux par rapport aux triplets disponibles. Si tous les triplets ont été utilisés et que les conditions de détection n'ont pas été modifiées (observation statique) alors l'algorithme a fait de son mieux. Il ne peut ajouter aucune information supplémentaire et est donc incapable de progresser davantage vers les objectifs. Il faut donc soit attendre en espérant un changement (comme du mouvement) soit terminer l'algorithme (en gardant à l'esprit que la terminaison n'est pas idéale). Le choix sera dépendant de l'application. Globalement, ce cas peut arriver parce que les détecteurs ne sont pas assez fiables ou pas assez précis pour atteindre les objectifs et ce malgré la fusion d'information qui permet d'être, au final, plus précis que chaque détecteur (ayant réussi la détection) pris individuellement.

Un cas particulier peut survenir lorsque la confiance initiale dans le vecteur de caractéristiques initial n'est pas totale (différente de 1), comme par exemple pour une application de reconnaissance d'objets dans une image donnée pour laquelle l'objet est potentiellement présent (et donc potentiellement absent). Dans ce cas là, si toutes les premières détections échouent, il n'est sans doute pas nécessaire de poursuivre en appelant encore d'autres triplets puisque la cause la plus probable de ce cumul d'échecs est sûrement liée à la non présence de l'objet. L'algorithme pourra se terminer s'il est suffisamment sûr que l'objet n'était pas présent. Donc quand la confiance (ré-estimée) dans le vecteur de caractéristiques initial sera suffisamment faible. En effet, il est certain que l'objectif de précision ne pourra pas être atteint puisque l'objet n'est pas présent. L'algorithme se terminera si :

$$\mathrm{P}(\mathbf{E}_{0|0}|\overline{d_1},\overline{d_2},...) \leq 1-\mathrm{P}_{Obj}$$

III.8 Initialisation détaillée

Grâce à la description détaillée des différentes étapes de l'algorithme dans ce chapitre, nous avons pu voir quelles étaient les informations nécessaires à son fonctionnement. Nous allons donc dans cette partie récapituler les différents éléments à initialiser dont nous avons besoin.

Nous avons tout d'abord les éléments liés à l'objet : il faut donner une valeur pour son état initial, une fonction d'évolution et la jacobienne associée.

— $\mathbf{E}_{0|0} = \{\underline{\mathrm{X}}_{0|0}, \mathbf{C}_{0|0}, \mathrm{P}(\mathbf{E}_{0|0})\}$

— $\mathbf{f}$: fonction d'évolution du vecteur de caractéristiques $\underline{\mathrm{X}}$

— $\mathbf{F}$: jacobienne associée à $\mathbf{f}$. $\mathbf{f}$ et $\mathbf{F}$ sont utiles pour la prédiction de l'état de l'objet à l'étape l connaissant l'étape $l-1$ (voir le § II.3.2.1.f et le § III.6.2).

Ensuite, il faut s'occuper des différentes parties de l'objet :

— $\mathbf{h}_i$: une fonction d'observation permettant de relier le vecteur de caractéristiques $\underline{\mathrm{X}}$ au vecteur de caractéristiques $\underline{\mathrm{Y}}^i$ pour chaque partie i (voir le § II.3.2.2).

— $\mathbf{H}_i$ jacobienne associée à $\mathbf{h}_i$

— $\mathrm{P}(\mathrm{A}^i_{0|0})$) : une probabilité d'existence initiale pour la partie i

Nous devons définir les objectifs à atteindre (voir le § III.3.2)

— $\mathbf{C}_{Obj}$: matrice objectif de précision

— P_{Obj} : objectif de confiance

Il nous faut également caractériser nos détecteurs (voir les § III.4.3, III.3.5, et III.5) :

— α_j : taux de faux négatifs pour le détecteur j

— β_j : taux de faux positifs pour le détecteur j

— $\mathrm{A}_{d_{ref_j}}$: aire de référence pour le détecteur j

— $t_{d_{ref_j}}$: temps moyen nécessaire pour traiter une zone dont l'aire est l'aire de référence $\mathrm{A}_{d_{ref_j}}$ pour le détecteur j.

— S : seuil pour considérer une primitive compatible. Le même seuil est utilisé pour tous les détecteurs.

Il ne manque plus que les caractéristiques de l'environnement :

— ϵ : probabilité de détecter la bonne primitive même si l'estimation n'est pas intègre (voir le § III.4.3)

— $\mathrm{D}z_p$: densité de primitives de type p. Elle permettra de calculer le nombre probable de primitives de ce type dans une zone donnée (voir le § III.4.7.4).

III.9 Bilan

Nous avons présenté en détails les différentes étapes de notre algorithme. Une première phase d'initialisation est nécessaire. Certaines des données à initialiser sont liées à la modélisation de l'objet : il faut notamment fournir l'état initial de l'objet, les liens existants entre son vecteur de caractéristiques et ceux de ses parties ou encore sa fonction d'évolution (et la jacobienne associée) nécessaire pour la prédiction. Le reste des informations sert essentiellement pour les probabilités, à la fois afin de permettre la sélection du meilleur triplet (en prenant par exemple en compte les imperfections de détections de chaque détecteur) et à la fois pour avoir une mise à jour cohérente de la confiance .

Une fois tous ces éléments disponibles, l'algorithme pourra commencer son processus composé de trois étapes principales : une étape de sélection du meilleur triplet, une étape de détection et une étape de mise à jour. La première étape permet de choisir le triplet qui permettra le plus probablement de progresser au mieux vers les objectifs. Il s'agit de ne pas choisir au hasard et de prendre en compte les informations disponibles (comme les détections déjà réussies ou encore les caractéristiques des différents détecteurs) pour réaliser un choix le plus judicieusement possible.

Une fois le meilleur triplet choisi, un processus de détection est exécuté et profite des procédés de focalisation. En particulier, la détection est le plus souvent faite dans une région restreinte des données capteurs grâce à la *focalisation spatiale*. En outre, la *focalisation de caractéristiques* permet de ne sélectionner que les primitives compatibles avec la partie cherchée. L'état de la partie cherchée est mis à jour lors de ce procédé de détection.

Une fois la détection achevée, la mise à jour de l'état de l'objet est effectuée. Une mise à jour de la confiance est toujours faite à l'aide du réseau bayésien. En cas de réussite de la détection, une mise à jour de l'estimation est possible grâce à un filtre de Kalman. En cas d'échec de la détection, un processus de remise en cause sera éventuellement effectué dans l'espoir d'annuler une mauvaise association faite lors d'une étape précédente. Un test est donc fait pour savoir si le retour est nécessaire et si oui, l'itération de retour est déterminée et la confiance sera recalculée.

Enfin, un critère de fin sera utilisé à l'issue de ces trois étapes pour déterminer la terminaison de l'algorithme. S'il doit se poursuivre, les trois étapes précédentes seront à nouveau réalisées.

Le fonctionnement théorique de l'algorithme a donc été décrit dans ce chapitre. Nous allons, dans le chapitre suivant, présenter quelques applications de nature différente afin d'illustrer les processus évoqués dans ce chapitre et de montrer la validité de notre algorithme ainsi que sa généricité.

Chapitre IV

Expérimentations et Résultats

« Le monde que nous avons créé est le résultat de notre niveau de réflexion, mais les problèmes qu'il engendre ne sauraient être résolus à ce même niveau. »

Albert Einstein

Sommaire

IV.1 Introduction

Lors des chapitres précédents, nous avons proposé un algorithme censé pouvoir s'appliquer à différentes situations et dans différents contextes. Nous allons maintenant présenter des applications variées afin de démontrer sa généricité. La plupart seront dans le domaine de la reconnaissance d'objets, identifiés à l'avance, mais nous présenterons aussi des résultats pour une application de localisation de robot mobile.

IV.2 Reconnaissance de polygones à M sommets

La première application proposée a une vocation pédagogique mais peut présenter un intérêt pour la reconnaissance d'objets manufacturés. L'objectif est de trouver, dans une image, un polygone à M sommets, comme ceux présentés dans la figure IV.1. Les segments du polygone pourront s'intersecter entre eux sans que cela ne modifie le nombre de sommets considérés.

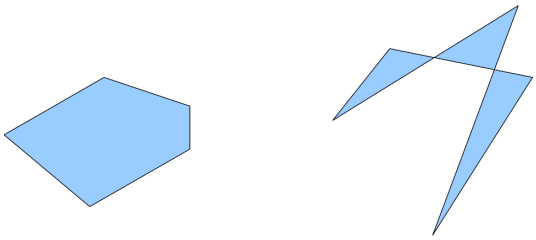


FIGURE IV.1 – deux exemples de polygones à 5 sommets.

IV.2.1 Modélisation du problème

IV.2.1.1 Modélisation de l'objet

L'objet sera représenté par le vecteur des M sommets qui le compose :

$$\underline{\mathrm{X}} = \begin{pmatrix} u_1 & v_1 & u_2 & v_2 & \dots & u_\mathrm{M} & v_\mathrm{M} \end{pmatrix}^\mathrm{T}$$

où (u_i, v_i) sont les coordonnées d'un sommet.

IV.2.1.2 Modélisation des parties

Chaque partie représentera par exemple ici, un segment du polygone. Elles seront donc de la forme :

$$\forall\ i \in [1, \mathrm{M}-1],\ \underline{\mathrm{Y}}^i = \begin{pmatrix} u_i & v_i & u_{i+1} & v_{i+1} \end{pmatrix}^\mathrm{T}$$

et pour la dernière partie :

$$\underline{\mathrm{Y}}^\mathrm{M} = \begin{pmatrix} u_\mathrm{M} & v_\mathrm{M} & u_1 & v_1 \end{pmatrix}^\mathrm{T}$$

Pour rappel, $\mathbf{h}_i$ est la fonction permettant de relier le vecteur de caractéristiques de la partie $\underline{\mathrm{Y}}^i$ au vecteur de caractéristiques de l'objet $\underline{\mathrm{X}}$: $\underline{\mathrm{Y}}^i = \mathbf{h}_i(\underline{\mathrm{X}})$ (voir le § II.3.2.2). Avec la modélisation proposée et en prenant pour convention que les indices de matrice débutent à 1, $\mathbf{h}_i$ peut donc être définie comme suit :

$$\forall\, i \in [1, \mathrm{M}-1], \mathbf{h}_i(\underline{\mathrm{X}}) = \begin{pmatrix} 0 & 0 & \dots & \overset{2i}{\overset{\downarrow}{1}} & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \end{pmatrix} \underline{\mathrm{X}}$$

et

$$\mathbf{h}_{\mathrm{M}}(\underline{\mathrm{X}}) = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \end{pmatrix} \underline{\mathrm{X}}$$

Pour la fonction d'évolution de l'objet $\mathbf{f}_i$ (voir le § II.3.2.1.f), nous prendrons la fonction identité. Entre deux itérations de l'algorithme, l'image traitée sera la même. En conséquence, le polygone cherché sera à l'identique.

Les autres éléments nécessaires à l'initialisation seront présentés dans le § IV.2.2.2.a .

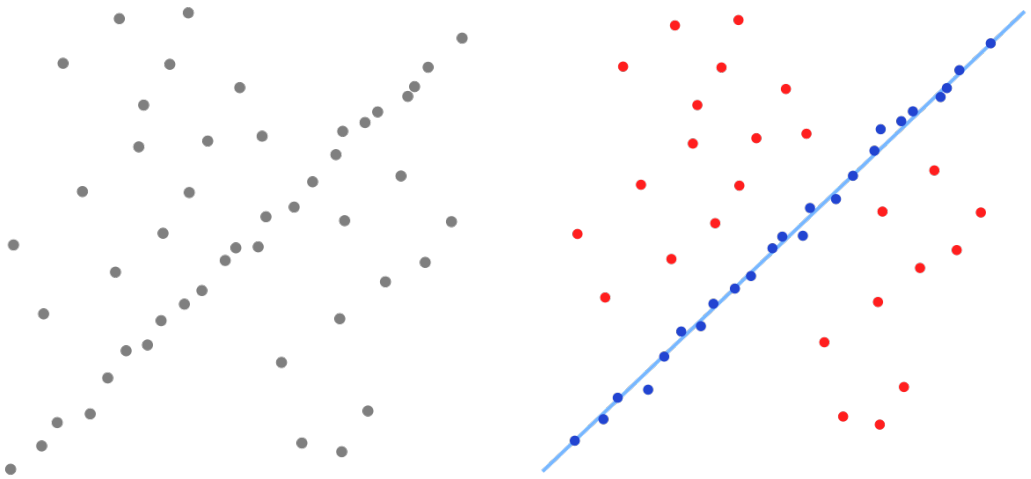
IV.2.2 Résultats et comparaison avec l'algorithme du RANSAC

Nous avons vu précédemment que, grâce à la détection des différentes parties qui composent un objet, nous pouvions estimer le vecteur de caractéristiques de l'objet. Afin d'avoir une bonne estimation, il est nécessaire de faire les bonnes associations entre les primitives renvoyées par les détecteurs et les parties constituant l'objet. D'une certaine manière, avec notre algorithme, la détection de l'objet peut être vue comme un problème d'association.

Nous souhaitons confronter notre algorithme (abrégé AF pour Algorithme Focalisant par la suite) à des approches plus traditionnelles d'association de données comme l'algorithme du RANSAC [Fischler and Bolles, 1981]. Nous allons donc expliquer le principe de fonctionnement de ce dernier. Ensuite, nous étudierons les résultats obtenus par l'AF et le RANSAC pour la détection d'un polygone à M sommets.

IV.2.2.1 Présentation de l'algorithme du RANSAC

Le principe de fonctionnement de l'algorithme du RANSAC [Fischler and Bolles, 1981] est le suivant : à partir d'un ensemble de primitives déjà identifiées, comme des points, un sous échantillon de ces primitives est sélectionné aléatoirement. Puis un modèle est créé et évalué à l'aide notamment des données restantes. Ceci est répété plusieurs fois dans l'espoir d'obtenir à la fin un "bon" modèle. Un exemple très connu d'utilisation de l'algorithme du RANSAC est l'estimation des paramètres d'une ligne passant au mieux parmi un ensemble de points dont certains sont des points aberrants (qui n'appartiennent pas à la ligne cherchée), voir la figure IV.2. À partir de deux points, il est possible de créer une ligne, les points sont alors triés entre inliers (points appartenant à la ligne) et outliers (points n'appartenant pas à la ligne). S'il y a suffisamment d'inliers, une erreur (somme des distances entre les points inliers et la ligne) est calculée et selon la valeur de cette erreur (et la valeur des erreurs des modèles précédemment testés), la ligne calculée sera conservée.



(a) exemple de données brutes à traiter (b) exemple de ligne renvoyée par le RANSAC

FIGURE IV.2 – Illustration de l'algorithme du RANSAC : à gauche les données brutes, à droites le résultat rendu par l'algorithme du RANSAC

L'algorithme du RANSAC ou une de ses variantes sert souvent dans l'estimation de pose. En effet, un des avantages de cet algorithme est qu'il est peu sensible au bruit. Il est la plupart du temps capable de trouver le modèle avec une bonne précision même s'il y a cinquante pour cent d'outliers. Il est donc particulièrement robuste. Il est aussi dans sa conception très générique. Si un modèle peut être construit à partir d'un sous ensemble de primitives et qu'il est possible d'estimer une erreur entre le modèle proposé et les données, l'algorithme du RANSAC pourra être utilisé. Par contre, le RANSAC nécessite de fixer des seuils en fonction de l'application visée (pour déterminer le nombre maximum d'itérations, le nombre ou le taux minimum d'inliers nécessaire pour considérer le modèle viable etc). Un autre défaut de l'algorithme du RANSAC provient de sa nature aléatoire. Si elle lui permet de se séparer des outliers, elle a l'inconvénient de ne pas permettre de garantir que d'un appel sur l'autre, avec les mêmes données, le résultat sera identique. En cas de multiples objets similaires présents dans l'image, il n'est pas non plus garanti que l'algorithme du RANSAC parvienne à en trouver un seul. En effet les autres objets ne sont pas tout à fait du bruit puisqu'ils ont leur propre cohérence mais sont néanmoins des outliers les uns par rapport aux autres. Cela peut faire osciller l'algorithme du RANSAC entre des modèles qui ne représentent pas le même objet et qui se retrouvent pourtant en concurrence.

Pour pallier les défauts du RANSAC ou le rendre encore plus performant, il existe une multitude de variantes. Le DEGENSAC [Chum et al., 2005] permet de le rendre plus fiable en lui permettant d'éliminer les cas dégénérés. Le LO-RANSAC [Chum et al., 2003] fait une optimisation locale sur le set d'inliers afin de respecter la convergence vers la solution. Le MAC-RANSAC [Rabin et al., 2010] quant à lui, permet de gérer le cas de multiples objets du même type dans une image. Mais les approches dérivées de la version initiale cherchent surtout à optimiser les temps de calcul pour atteindre la solution finale. Le PROSAC[Chum and Matas, 2005] sélectionne prioritairement les candidats les plus prometteurs car plus ressemblants aux points d'origine. Le SCRAMSAC [Sattler et al., 2009] va exiger une certaine consistance des données dans un voisinage afin d'éliminer les solutions non consistantes plus rapidement.

Nous avons choisi de comparer l'AF au RANSAC car ses variantes sont souvent teintée "calcul de pose" ou utilisent des heuristiques dépendantes des données alors que nous cherchons à conserver la généricité.

IV.2.2.2 Comparaison avec le RANSAC avec contrainte d'ordre

IV.2.2.2.a Conditions de tests

Pour l'initialisation (voir le § III.8 pour un récapitulatif de tous les éléments nécessaires à cette dernière), nous considérons que les segments du polygone auront pour seule contrainte d'être entièrement dans l'image. Ainsi, le vecteur de caractéristiques initial $\underline{X}_{0|0}$ est :

$$\underline{X}_{0|0} = \left(\frac{width}{2} \quad \frac{height}{2} \quad \dots \quad \frac{width}{2} \quad \frac{height}{2} \right)^{T}$$

où $width$ est la largeur de l'image et $height$ sa hauteur. La matrice de covariance initiale associée, $\mathbf{C}_{0|0}$ est :

$$\mathbf{C}_{0|0} = \begin{pmatrix} \left(\frac{width}{2}\right)^2 & 0 & \dots & 0 & 0 \\ 0 & \left(\frac{height}{2}\right)^2 & \dots & 0 & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \left(\frac{width}{2}\right)^2 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{height}{2}\right)^2 \end{pmatrix}$$

La confiance initiale choisie est : $P(\mathbf{E}_{0|0}) = 1$. Nous considérons ainsi qu'il y a toujours un polygone dans l'image. De plus, tous les segments du polygone doivent être présents donc $P(A^i_{0|0}) = 1, \ \forall i \in [1, M]$.

FIGURE IV.3 – Deux exemples de scénarios pour des polygones à 4 sommets. En vert le polygone à trouver, en rose d'autres lignes présentes (bruit).

Les tests seront réalisés sur des données de synthèse comme celles proposées dans la figure IV.3. Ces scénarios synthétiques ont été créés de manière automatique et aléatoire en respectant les contraintes fixées. Dans chaque scénario, il y a un polygone de taille connue (le nombre de sommets est fixé à l'avance) ainsi qu'un ensemble d'autres segments. Pour des raisons de visualisation, les polygones recherchés ont été dessinés en vert dans la

figure IV.3 alors que les autres segments apparaissent en rose, néanmoins la couleur ne sera pas prise en compte dans cette étude.

Le détecteur de segments utilisé est supposé parfaitement fiable ($\alpha = 0$ et $\beta = 0$). Il renvoie tous les segments et ce même si en pratique ils se croisent ou seraient difficiles à identifier. Le détecteur est compatible avec toutes les parties définies précédemment. Pour l'AF, la précision du détecteur est fixée à 3 pixels sur chaque coordonnée. Tous les segments ont été prédétectés, ainsi les deux algorithmes auront à leur disposition exactement la même liste de primitives. Cette hypothèse est en défaveur de notre algorithme qui ne pourra pas profiter de l'optimisation des temps de détection liée à la focalisation spatiale.

Le choix du triplet va donc se résumer au choix de la partie : il y a un seul détecteur disponible (le détecteur défini précédemment) et un seul capteur possible, il est supposé avoir renvoyé les images du polygone.

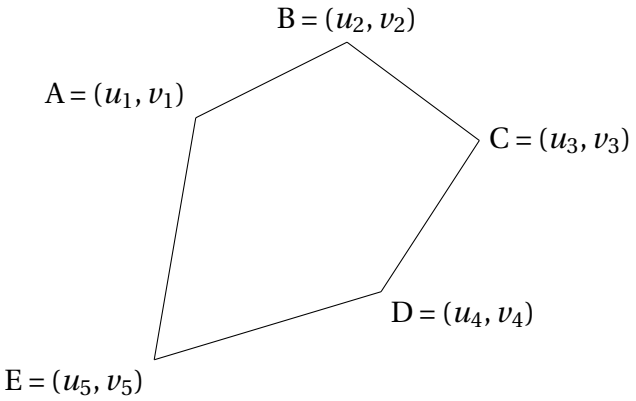


FIGURE IV.4 – Polygone à 5 sommets avec affichage des paramètres de ses sommets.

Une exigence fixée est que les segments doivent être trouvés **dans le bon ordre**. Nous abrégerons le vecteur $\begin{pmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4 & u_5 & v_5 \end{pmatrix}^T$ en $\begin{pmatrix} A & B & C & D & E \end{pmatrix}^T$ et similairement pour d'autres vecteurs de caractéristiques. Pour le polygone défini dans la figure IV.4 :

$$\begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix}, \begin{pmatrix} A \\ E \\ D \\ C \\ B \end{pmatrix}, \begin{pmatrix} B \\ C \\ D \\ E \\ A \end{pmatrix}, \begin{pmatrix} B \\ A \\ E \\ D \\ C \end{pmatrix}, \begin{pmatrix} C \\ D \\ E \\ A \\ B \end{pmatrix}, \begin{pmatrix} C \\ B \\ A \\ E \\ D \end{pmatrix}, \begin{pmatrix} D \\ E \\ A \\ B \\ C \end{pmatrix}, \begin{pmatrix} D \\ C \\ B \\ A \\ E \end{pmatrix}, \begin{pmatrix} E \\ A \\ B \\ C \\ D \end{pmatrix}, \begin{pmatrix} E \\ D \\ C \\ B \\ A \end{pmatrix}$$

est l'ensemble des vecteurs acceptables.

$$\begin{pmatrix} A \\ C \\ B \\ D \\ E \end{pmatrix}, \begin{pmatrix} B \\ D \\ E \\ C \\ A \end{pmatrix}, \begin{pmatrix} D \\ E \\ B \\ A \\ C \end{pmatrix}, \ldots$$

sont des exemples de vecteurs non acceptables.

Nous noterons que pour le RANSAC, cela signifie que la construction du modèle se fait de manière très simple : M segments sont tirés aléatoirement pour un polygone à M sommets. Le premier segment tiré est affecté à la première partie, le deuxième segment à la deuxième partie et ainsi de suite. Il serait possible d'optimiser le modèle construit par exemple en cherchant à minimiser la somme des distances entre les extrémités de deux parties consécutives.

Pour le critère de fin, les deux algorithmes termineront lorsqu'ils auront trouvé le polygone recherché (nous disposons du "vrai" polygone qui sert de référence) ou lorsqu'un délai de 3 secondes s'est écoulé.

Pour chaque scénario de test, chacune des méthodes renvoie un booléen indiquant si elle a trouvé le polygone recherché ainsi que le temps mis pour réussir la détection globale. Les temps des détecteurs ne sont pas pris en compte dans le total des temps car nous considérons que les détections ont déjà été réalisées en préambule du test.

Pour le comparatif, nous allons faire varier le nombre de sommets constituant le polygone recherché et le bruit dans l'image. Nous définissons par bruit le rapport entre le nombre de segments présents dans l'image ne faisant pas partie du polygone recherché (outliers) et le nombre total de segments présents dans le scénario (les M segments du polygone (inliers) additionnés des segments libres). Le bruit B est défini par :

$$B = \frac{s}{M + s}$$

où

— M est le nombre de sommets du polygone

— s est le nombre de segments n'appartenant pas au polygone.

Ainsi un bruit de 0% indique que dans l'image seuls sont présents les segments du polygone tandis qu'une valeur de 50% caractérise le fait qu'il y aura autant d'inliers que d'outliers.

Pour chaque couple de paramètres (nombre de sommets, pourcentage de bruit), 100 scénarios seront testés. Ainsi le pourcentage de réussite de chacun des deux algorithmes sera calculé statistiquement.

IV.2.2.2.b Comparaison sans bruit

Nous considérons ici, le cas où tous les segments appartiennent au polygone recherché. Il n'y a pas d'outliers. La figure IV.5 montre que pour des polygones avec peu de sommets, le RANSAC est plus intéressant en temps de calcul mais qu'au delà de 8 sommets, il devient plus gourmand que l'AF. Plus le nombre de sommets augmente, plus le temps de traitement des deux algorithme croît. Néanmoins la progression en temps de calcul de l'AF est plus lente que celle du RANSAC. Ceci lui permet, alors qu'il commence avec des temps plus élevés, de devenir plus intéressant.

En effet, les temps de calculs pour l'AF sont principalement liés au temps pris par les réseaux bayésiens pour calculer les probabilités. Le nombre de réseaux utilisés à chaque itération est dépendant du nombre de triplets et donc du nombre de sommets. À la première itération, l'AF choisira un segment pour une partie i. Pour les itérations suivantes, la partie à rechercher sera toujours une partie pour laquelle une des extrémités est fixée par

les détections précédentes (grâce au critère de sélection, voir la section III.3). Comme il n'y a pas de bruit, en choisissant une telle partie, l'AF fait nécessairement la bonne affectation.

À l'inverse, pour le RANSAC, la combinatoire est en sa défaveur. Pour un polygone à M sommets, dans le cas sans bruit, l'affectation de la première partie sera toujours la bonne (le sommet de départ du polygone est considéré sans importance). Pour la seconde partie, il y a par contre $\frac{2}{n-1}$ chances de choisir la bonne primitive car le sens de parcours du polygone est sans importance (pour le polygone de la figure IV.4, si le premier segment choisi est AB, le deuxième peut au choix être BC ou AE). Pour la troisième partie, il y aura $\frac{1}{n-2}$ chances de faire la bonne association et ainsi de suite. Au final, à chaque itération, le RANSAC a $\frac{2}{(n-1)!}$ chances d'avoir réalisé toutes les bonnes associations et donc d'avoir réussi la détection du polygone recherché. Pour 3 sommets, dès la première itération, le polygone sera trouvé. Par contre, pour 9 sommets, il aura par itération $\frac{1}{20160}$ chances de trouver le polygone. C'est ce qui explique l'explosion des temps pour le RANSAC.

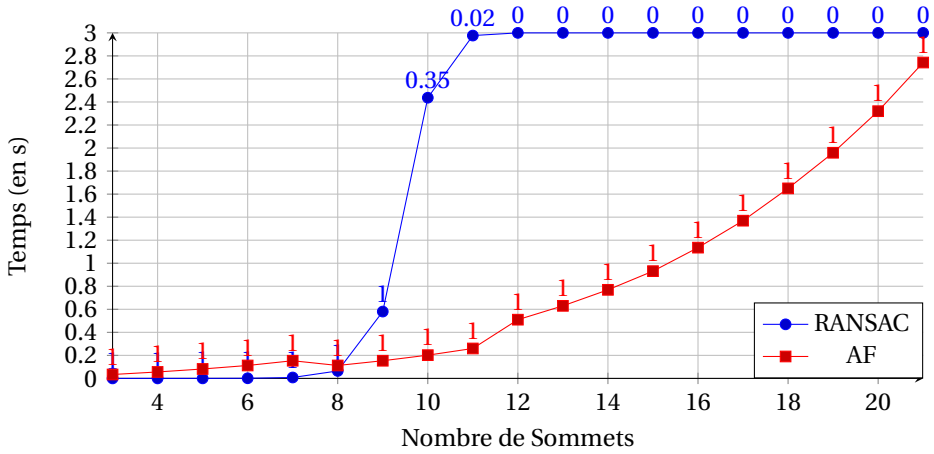


FIGURE IV.5 – Comparaison des performances de l'AF et du RANSAC. l'AF est représenté par la courbe rouge et le RANSAC par la courbe bleue. Le bruit est nul : dans tous les scenarii, les seuls segments présents sont ceux du polygone recherché. Le paramètre variable est le nombre de sommets du polygone. Le nombre au dessus de chaque point représente le pourcentage de réussite de chaque algorithme : ainsi une valeur de 1 indique que 100% des scenarii ont abouti à la détection du polygone cherché tandis qu'une valeur de 0.35 signifiera que l'algorithme considéré a réussi la détection pour seulement 35% des cas.

IV.2.2.2.c Comparaison avec bruit

Maintenant l'influence du bruit va être étudiée pour les cas où le RANSAC est meilleur que l'AF sans bruit, c'est à dire lorsque le polygone est composé de 3 à 8 sommets. Les figures IV.6, IV.7, IV.8, IV.9, IV.10, IV.11, montrent les performances des deux algorithmes pour respectivement 3, 4, 5, 6, 7 et 8 sommets avec un bruit variant de 0% à 90%. Plus le nombre de sommets est élevé et plus le bruit est important, plus vite l'AF devient intéressant en terme de temps de calcul.

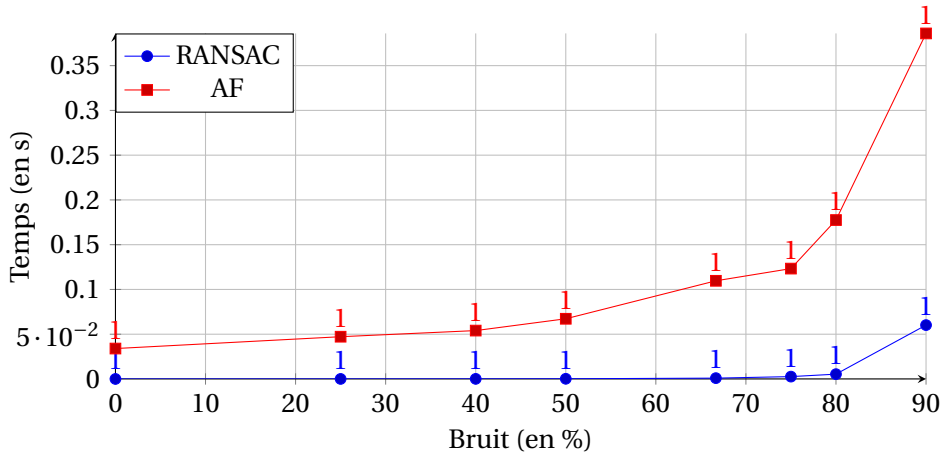


FIGURE IV.6 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 3. Le paramètre variable est le pourcentage de bruit.

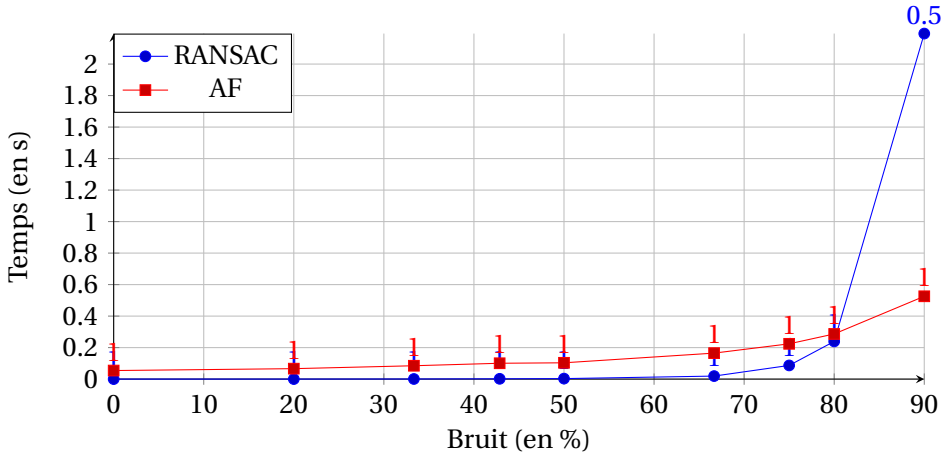


FIGURE IV.7 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 4. Le paramètre variable est le pourcentage de bruit.

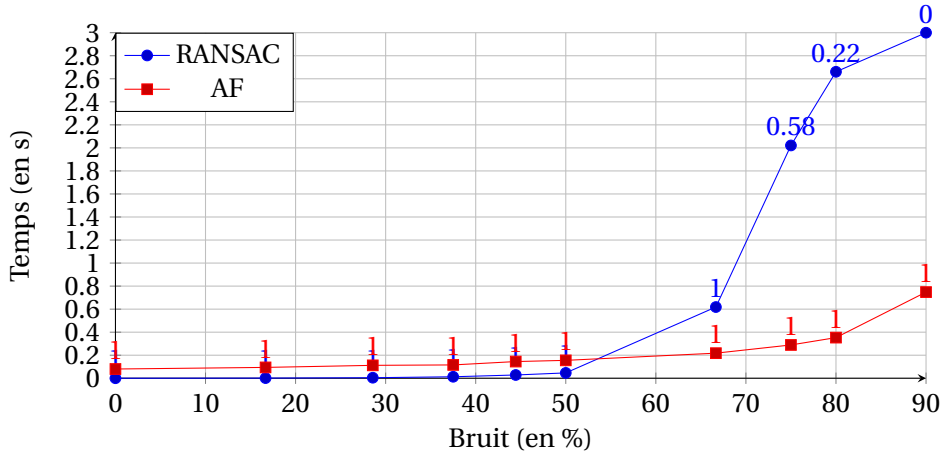


FIGURE IV.8 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 5. Le paramètre variable est le pourcentage de bruit.

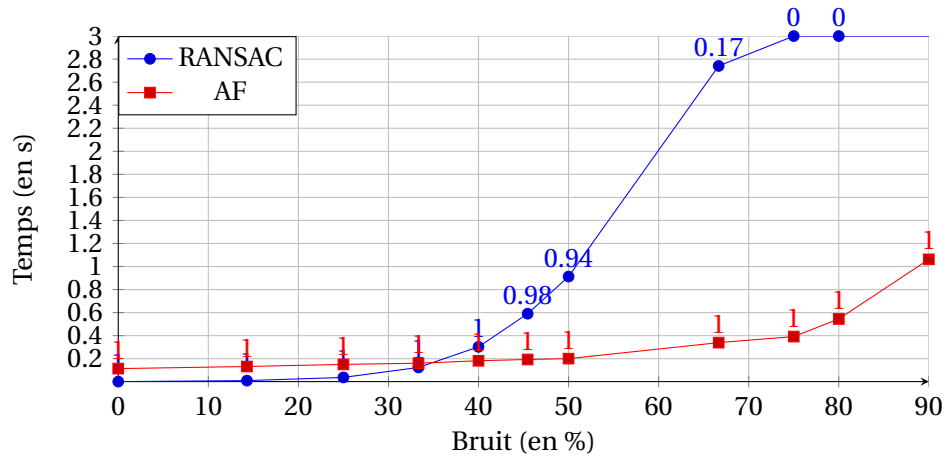


FIGURE IV.9 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 6. Le paramètre variable est le pourcentage de bruit.

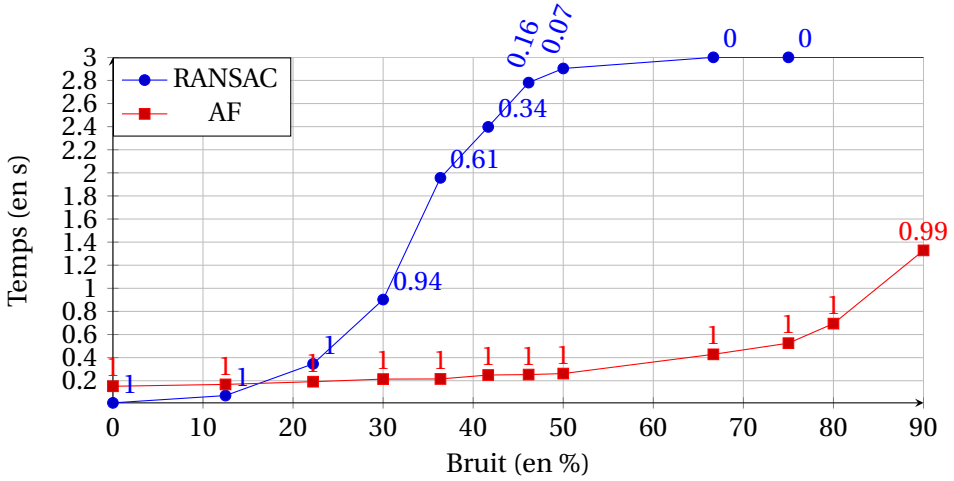


FIGURE IV.10 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 7. Le paramètre variable est le pourcentage de bruit.

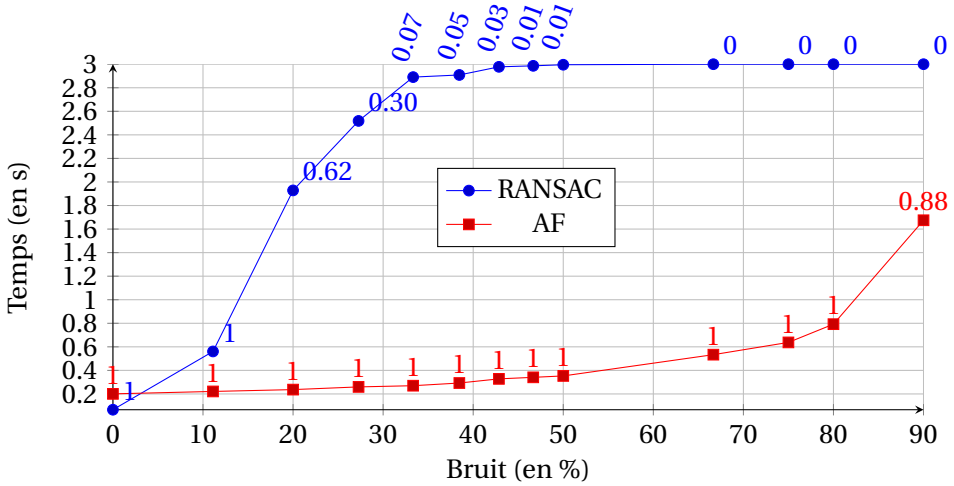


FIGURE IV.11 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 8. Le paramètre variable est le pourcentage de bruit.

IV.2.2.2.d Tableau récapitulatif

Le tableau IV.1 permet de résumer en fonction de la situation quel est l'algorithme le plus pertinent.

TABLEAU IV.1 – Tableau récapitulatif pour le comparatif de performances entre le RANSAC et l'AF en fonction du nombre de sommets et du bruit. Dans chaque case, R indique que c'est le RANSAC le meilleur en temps de calcul pour le cas considéré. AF indique lorsque notre algorithme est le plus performant.

Bruit	Nombres de Sommets						
	3	4	5	6	7	8	≥9
0%	R	R	R	R	R	R	AF
≥5%	R	R	R	R	R	AF	AF
≥23%	R	R	R	R	AF	AF	AF
≥40%	R	R	R	AF	AF	AF	AF
≥53%	R	R	AF	AF	AF	AF	AF
≥80%	R	AF	AF	AF	AF	AF	AF

IV.2.2.2.e Conclusion

Les résultats observés correspondent à l'idée que l'*intelligence* de l'algorithme AF a un coût (provenant entre autres des calculs du critère, des probabilités, etc) qui ne devient rentable que lorsque le problème se complexifie : soit car le nombre de paramètres à prendre en compte augmente (en fonction du nombre de sommets), soit car le bruit devient plus conséquent ce qui perturbe un système purement bottom-up (comme le RANSAC). En effet, du fait des focalisations effectuées par l'AF, seule une *faible* partie du bruit a des conséquences : une fois un segment du polygone trouvé, seuls les segments dont les extrémités sont assez proches de ce segment sont traités.

IV.2.2.3 Comparaison avec le RANSAC sans la contrainte d'ordre

IV.2.2.3.a Modification des conditions de tests

Cette fois-ci, la contrainte d'ordre est éliminée pour le RANSAC : le polygone est supposé être reconstruit lorsque ses M segments sont piochés quel que soit leur ordre. Lorsqu'il n'y a pas de bruit, le RANSAC est par conséquent absolument certain de trouver dès la première itération le bon polygone : il ne peut pas faire de mauvaise association.

Pour l'AF, le calcul du réseau bayésien était fait précédemment à l'aide de la bibliothèque Dlib [King, 2009]. Les calculs seront désormais réalisés avec la méthode proposée dans l'annexe C au § C.4. Cette dernière permet d'optimiser les calculs dans le cas de réseaux bayésiens avec des événements binaires et dont la structure est fixée. C'est le cas pour nos applications où la structure du réseau bayésien est constante et ce sont les valeurs des tables de probabilités conditionnelles qui peuvent varier. De plus, nous nous intéressons seulement aux probabilités d'un ensemble restreint de nœuds. La méthode C.4 est bien adaptée dans ce cas : son premier appel déterminera quelles cases de la table construite sont utiles et ensuite seules les valeurs de ces cases seront utilisées.

Nous testerons également l'influence de la précision estimée du détecteur sur les temps de calculs pour l'AF.

IV.2.2.3.b Comparaison et remarques

La figure IV.12 montre qu'effectivement lorsqu'il n'y a pas de bruit, sans la condition sur l'ordre, le RANSAC devient beaucoup plus intéressant. Les temps de l'AF dépendent eux du nombre de sommets : à chaque itération, pour chaque triplet (donc ici pour chaque sommet), un réseau bayésien est calculé pour savoir quel segment est le plus adéquat à détecter en premier. Or quand il n'y a aucun bruit, ce calcul est fastidieux et n'apporte rien.

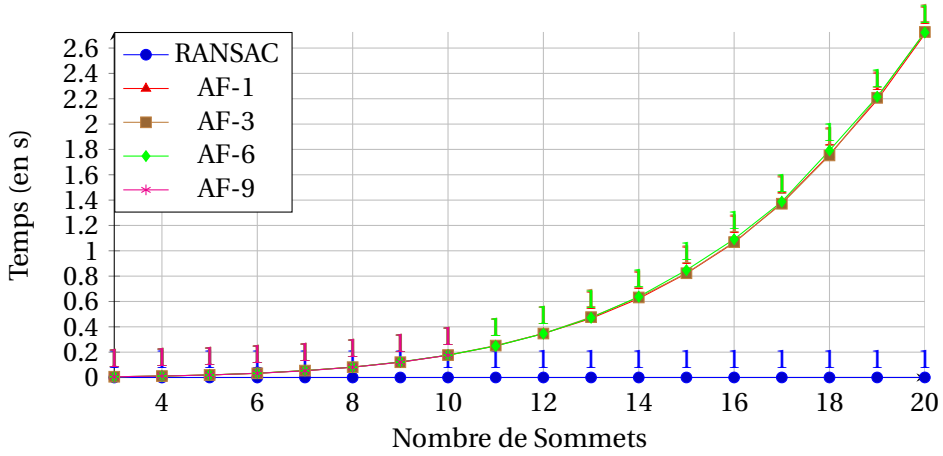


FIGURE IV.12 – Comparaison des performances de l'AF et du RANSAC sans la contrainte d'ordre. Le RANSAC est toujours représenté par la courbe bleue. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels. Le bruit est nul : les seuls segments présents sont ceux du polygone recherché. Le paramètre variable est le nombre de sommets du polygone. La contrainte d'ordre a été supprimée pour le RANSAC.

La condition sur l'ordre facilite la tâche de reconnaissance pour le RANSAC tandis que les calculs plus adaptés pour les réseaux bayésiens diminuent grandement les temps de calculs pour l'AF. Avec ces nouvelles conditions, le même phénomène qu'auparavant peut être observé dans les figures IV.13, IV.14, IV.15, IV.16, IV.17, IV.18 : pour les scénarios avec peu de sommets et peu de bruit, le RANSAC est le plus pertinent car l'*intelligence* de l'AF a un coût trop prohibitif pour être rentable. Par contre, l'AF est moins sensible au bruit lorsque celui-ci est important. En effet, du fait de sa focalisation de caractéristiques, l'AF est capable d'éliminer plus efficacement les mauvais segments.

Le dernier point mis en exergue dans les graphiques IV.13, IV.14, IV.15, IV.16, IV.17, IV.18 est l'influence de la précision du détecteur sur les temps de calculs. En effet, moins le détecteur est précis, plus il sera facile de trouver des primitives compatibles. Selon la précision, l'extrémité d'un nouveau segment cherché devra être dans un cercle de 1, 3, 6 ou 9 pixels autour d'une des extrémités déjà connues. Donc plus cette valeur est élevée, plus le cercle est grand et plus il y a de chances qu'un des segments parmi les outliers ait une des ses extrémités dans le cercle. Ce segment, s'il existe, sera donc une primitive compatible et pourra donner lieu à une mauvaise association qui nécessitera un retour (et donc des temps de calculs supplémentaires pour trouver la bonne solution). Plus le bruit est élevé, plus il y a de segments outliers et donc il devient de plus en plus probable que l'un d'entre eux ait son extrémité suffisamment proche d'un segment appartenant au polygone recherché.

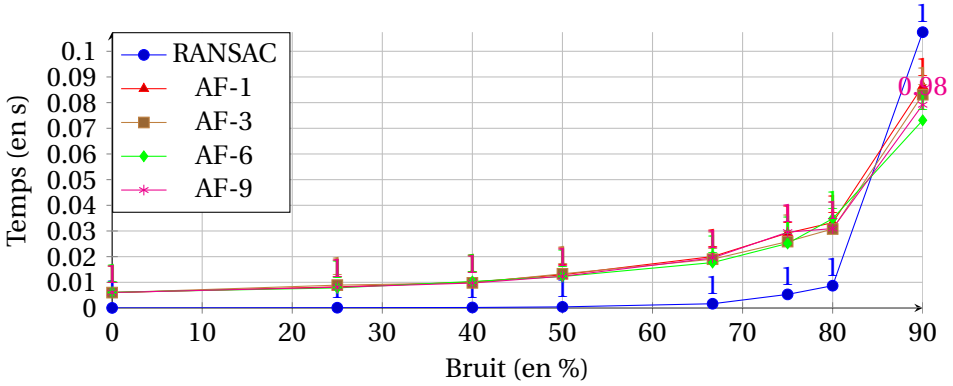


FIGURE IV.13 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 3. Le paramètre variable est le pourcentage de bruit. La contrainte d'ordre a été retirée pour le RANSAC. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels

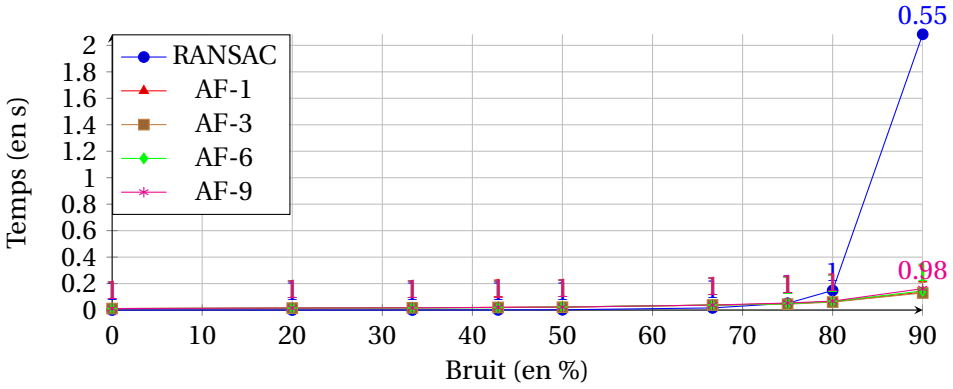


FIGURE IV.14 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 4. Le paramètre variable est le pourcentage de bruit. La contrainte d'ordre a été retirée pour le RANSAC. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels

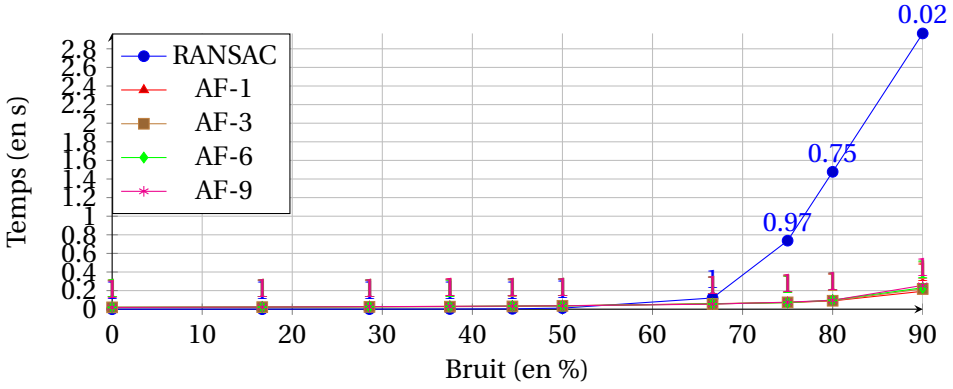


FIGURE IV.15 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 5. Le paramètre variable est le pourcentage de bruit. La contrainte d'ordre a été retirée pour le RANSAC. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels

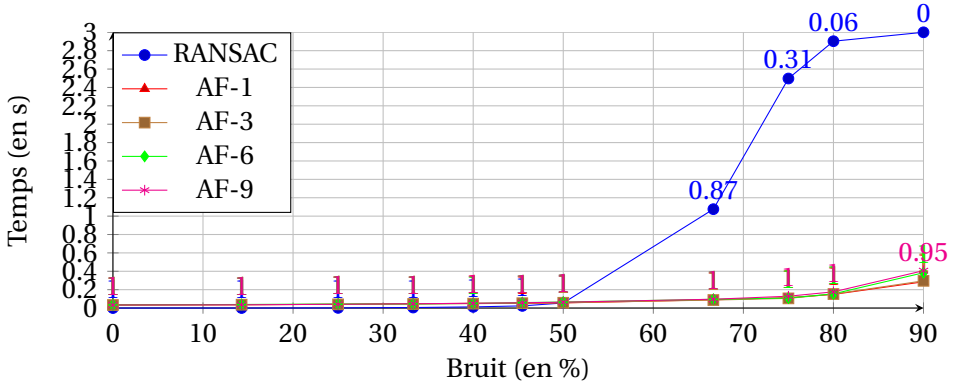


FIGURE IV.16 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 6. Le paramètre variable est le pourcentage de bruit. La contrainte d'ordre a été retirée pour le RANSAC. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels

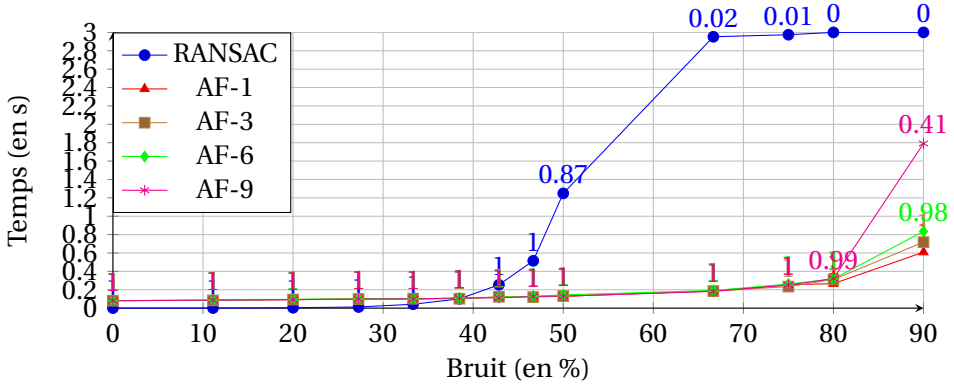


FIGURE IV.17 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 8. Le paramètre variable est le pourcentage de bruit. La contrainte d'ordre a été retirée pour le RANSAC. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels

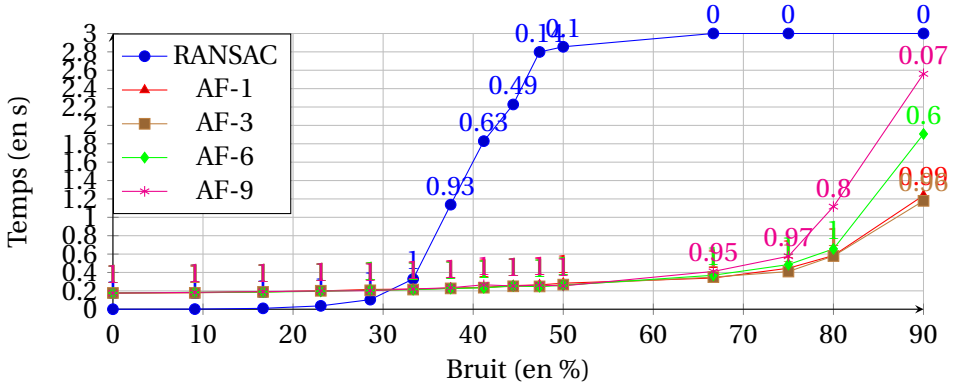


FIGURE IV.18 – Comparaison des performances de l'AF et du RANSAC. Le nombre de sommets du polygone recherché est fixé à 10. Le paramètre variable est le pourcentage de bruit. La contrainte d'ordre a été retirée pour le RANSAC. Les courbes rouge, marron, verte et rose correspondent respectivement aux résultats de l'AF lorsque la précision du détecteur est de 1, 3, 6 et 9 pixels

IV.2.3 Conclusion

Nous avons comparé notre AF avec le RANSAC dans différentes configurations. Le RANSAC est plus adapté pour des problèmes avec peu de paramètres et sans trop de bruit. L'AF grâce notamment à son système de focalisation spatiale est bien moins sensible au bruit. Par contre, l'*intelligence* nécessaire est coûteuse en terme de temps et ne devient rentable que lorsque la situation se complexifie. En outre, plus un détecteur est précis, moins l'AF est influencé par le bruit car la focalisation est plus efficace dans cette situation.

IV.3 Reconnaissance de panneaux de limitation de vitesse

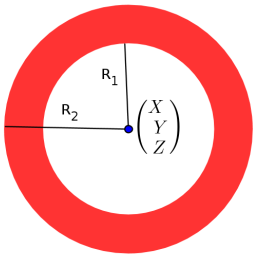
Nous proposons ici une application avec des objets relativement simples mais en travaillant avec des données réelles. L'objectif ici sera de détecter des panneaux de limitation de vitesse.

IV.3.1 Modélisation

IV.3.1.1 Modélisation de l'objet

L'objet sera modélisé par sa position dans l'espace $\begin{pmatrix} X & Y & Z \end{pmatrix}^T$, le rayon du cercle extérieur R_2 et le rapport des rayons $r = \frac{R_1}{R_2}$ avec R_1 le rayon du cercle intérieur (voir la figure IV.19b). La position du panneau est définie par rapport au centre de la caméra (voir la figure IV.19a). Le repère est tel que l'axe Z indique la distance entre le panneau et la caméra.

(a) position du panneau par rapport à la caméra



(b) représentation des paramètres du panneau

FIGURE IV.19 – Modélisation d'un panneau de limitation de vitesse.

Le vecteur de caractéristiques $\underline{X}$ peut se mettre sous la forme :

$$\underline{X} = \begin{pmatrix} X & Y & Z & R_2 & r \end{pmatrix}^T$$

IV.3.1.2 Modélisation des parties

Nous allons décomposer le panneau de limitation de vitesse en trois parties : une partie *blobRouge*, une partie *petitCercle* et une partie *grandCercle* (voir figure IV.20). La partie *blobRouge* nécessitera la détection d'une primitive ***zoneRouge*** modélisée à l'aide des paramètres suivants : (u_b, v_b) le centre de la zone dans le repère caméra, L la longueur de la zone (selon l'axe u) et l la largeur de la zone (selon l'axe v) :

$$\underline{Y}^{blobRouge} = \begin{pmatrix} u_b & v_b & L & l \end{pmatrix}^T$$

Les deux parties *petitCercle* et *grandCercle* utiliseront la même primitive ***cercle*** caractérisée par les paramètres : (u_c, v_c) le centre du cercle dans l'image, et r_i son rayon. Ainsi :

$$\underline{Y}^{grandCercle} = \begin{pmatrix} u_c & v_c & r_2 \end{pmatrix}^{\mathrm{T}}$$

et

$$\underline{Y}^{petitCercle} = \begin{pmatrix} u_c & v_c & r_1 \end{pmatrix}^{\mathrm{T}}$$

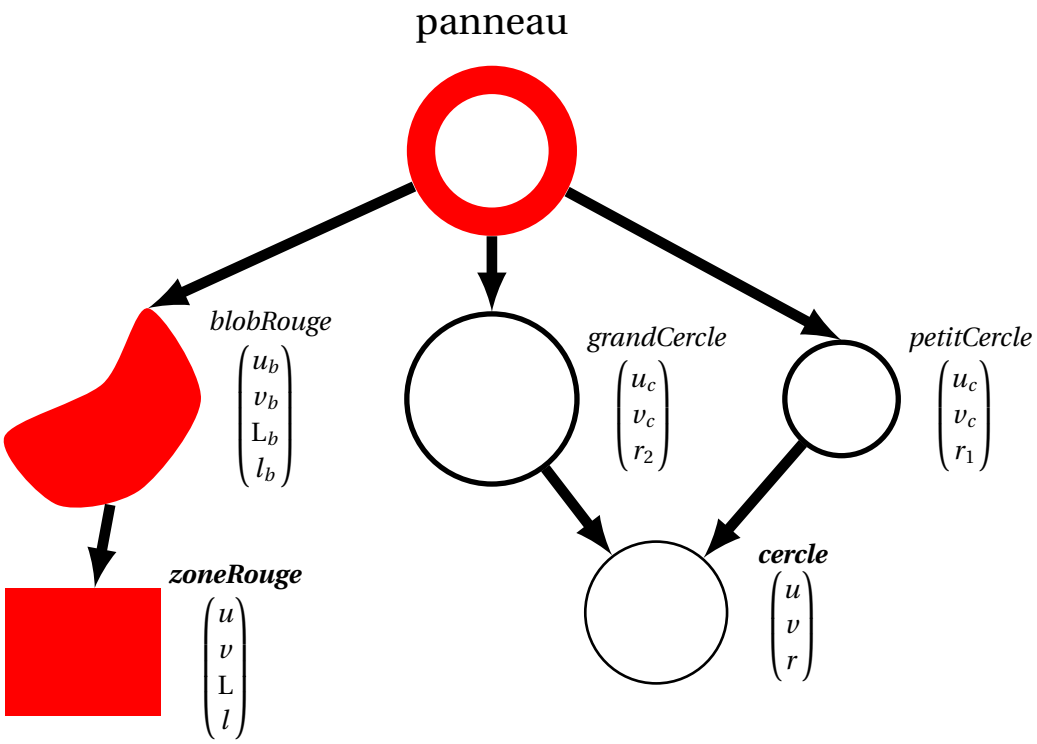


FIGURE IV.20 – Modélisation d'un panneau de limitation de vitesse.

Il faut remarquer que pour toutes les parties, le centre de la primitive correspondante sera le même : (u_c, v_c).

Nous allons maintenant montrer comment calculer les différents termes à partir du vecteur de caractéristiques de l'objet "panneau de limitation de vitesse". Nous faisons l'hypothèse que la caméra utilisée est calibrée et que nous disposons donc de sa matrice de calibration M_{calib} :

$$\mathrm{M}_{calib} = \begin{pmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \end{pmatrix}$$

IV.3.1.3 Lien entre le vecteur de caractéristiques des parties et celui de l'objet

IV.3.1.3.a Détermination du centre des parties (u_v, v_c)

Le centre du panneau en 3D est défini par $\begin{pmatrix} \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{pmatrix}^{\mathrm{T}}$. Il est donc possible de calculer sa projection en 2D :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} m11 & m12 & m13 & m14 \\ m21 & m22 & m23 & m24 \\ m31 & m32 & m33 & m34 \end{pmatrix} \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} m11\mathrm{X} + m12\mathrm{Y} + m13\mathrm{Z} + m14 \\ m21\mathrm{X} + m22\mathrm{Y} + m23\mathrm{Z} + m24 \\ m31\mathrm{X} + m32\mathrm{Y} + m33\mathrm{Z} + m34 \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \dfrac{m11\mathrm{X} + m12\mathrm{Y} + m13\mathrm{Z} + m14}{m31\mathrm{X} + m32\mathrm{Y} + m33\mathrm{Z} + m34} \\ \dfrac{m21\mathrm{X} + m22\mathrm{Y} + m23\mathrm{Z} + m24}{m31\mathrm{X} + m32\mathrm{Y} + m33\mathrm{Z} + m34} \end{pmatrix}$$

IV.3.1.3.b Détermination du rayon du grandCercle r_2

Pour déterminer, le rayon du grand cercle r_2, nous allons nous servir de la projection de quatre points 3D, définis comme suit :

$$\begin{aligned} \mathrm{P}_1 &= \begin{pmatrix} \mathrm{X} - \mathrm{R}_2 & \mathrm{Y} & \mathrm{Z} \end{pmatrix}^{\mathrm{T}} \\ \mathrm{P}_2 &= \begin{pmatrix} \mathrm{X} + \mathrm{R}_2 & \mathrm{Y} & \mathrm{Z} \end{pmatrix}^{\mathrm{T}} \\ \mathrm{P}_3 &= \begin{pmatrix} \mathrm{X} & \mathrm{Y} - \mathrm{R}_2 & \mathrm{Z} \end{pmatrix}^{\mathrm{T}} \\ \mathrm{P}_4 &= \begin{pmatrix} \mathrm{X} & \mathrm{Y} + \mathrm{R}_2 & \mathrm{Z} \end{pmatrix}^{\mathrm{T}} \end{aligned}$$

Leur projection en 2D est calculée selon le même principe que précédemment IV.3.1.3.a , ce qui donne donc quatre couples : $\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, \begin{pmatrix} u_3 \\ v_3 \end{pmatrix}, \begin{pmatrix} u_4 \\ v_4 \end{pmatrix}$.

En 3D, les points P_1 et P_2 sont à une distance de $2\,\mathrm{R}_2$, et il en va de même pour P_3 et P_4. L'hypothèse est faite qu'en 2D, la distance entre leur projection sera respectivement de r_{2_x} et r_{2_y} :

$$\begin{aligned} 2\,r_{2_x} &= \sqrt{(u1-u2)^2 + (v1-v2)^2} \\ 2\,r_{2_y} &= \sqrt{(u3-u4)^2 + (v3-v4)^2} \end{aligned}$$

Selon, l'angle de vue, c'est bien une ellipse qui est censée être observable dans l'image dont les demi-longueurs et les demi-largeurs des axes devraient valoir r_{2_x}, r_{2_y}. Mais, ici, il est supposé qu'un cercle sera conservé dans l'image (cette hypothèse était déjà présente implicitement dans la modélisation des parties puisque des primitives ***cercles*** avaient été choisies). Pour obtenir r_2, la moyenne entre r_{2_x} et r_{2_y} est utilisée :

$$\begin{aligned} r_2 &= \frac{r_{2_x} + r_{2_y}}{2} \\ &= \frac{\sqrt{(u1-u2)^2 + (v1-v2)^2} + \sqrt{(u3-u4)^2 + (v3-v4)^2}}{4} \end{aligned}$$

IV.3.1.3.c Détermination du rayon du petitCercle r_1

Pour le calcul de r_1, le même procédé que précédemment est utilisé mais en prenant $\mathrm{R}_1 = r\,\mathrm{R}_2$ à la place de R_2 lors de la définition des points et en supposant que dans l'image, les distances seront donc r_{1_x} et r_{1_y}.

IV.3.2 Résultats

Nous allons présenter ici quelques résultats obtenus avec l'approche que nous proposons pour la détection de panneaux de limitation de vitesse.

IV.3.2.1 Conditions de tests

Nous supposons avoir un vecteur initial $\underline{X}_{0|0}$ et une covariance initiale $\mathbf{C}_{0|0}$ cohérents avec la caméra utilisée et les données. Ils pourront avoir été obtenus par apprentissage sur différentes images. La probabilité $P(\mathbf{E}_{0|0})$ est initialisée à 0.5 : nous ignorons s'il y a ou non un panneau de limitation dans l'image traitée. Les probabilités $P(A^i_{0|0})$ valent 1 : toutes les parties du panneaux sont censées exister.

Nous supposons disposer d'un détecteur de ***zoneRouge*** peu précis mais très rapide. Notre détecteur de ***cercles*** sera, à l'inverse, précis mais très lent. Avec ces hypothèses, notre algorithme choisira toujours le détecteur de ***zoneRouge*** en premier.

IV.3.2.2 Illustration de la focalisation spatiale

Nous avons vu, au cours de ce manuscrit, que la focalisation spatiale avait de nombreux intérêts. Elle permet de diminuer les temps de calculs (en lançant les détecteurs uniquement dans une zone restreinte de l'image) ; nous avons montré les variations des temps de calculs dans l'application précédente voir section IV.2. Elle permet également d'augmenter la résistance au bruit (en augmentant le rapport signal sur bruit).

Nous allons ici illustrer ce phénomène de focalisation pour le panneau de limitation de vitesse en dessinant les zones de focalisation sur différentes images.

La figure IV.21 montre quelques exemples de ces différentes zones de recherche pour chaque partie : au fur et à mesure des illustrations, les zones se resserrent. Il est possible de remarquer qu'après la détection du *blobRouge*, les zones de focalisation associées aux parties *petitCercle* et *grandCercle* sont plus grandes que la zone représentant le *blobRouge* alors que les deux cercles devraient se trouver à l'intérieur. C'est en fait, un phénomène normal qui provient de l'imprécision associée au détecteur de ***zoneRouge***, la zone est plus grande pour pouvoir tolérer les approximations de détection.

IV.3.2.3 Évolution sur plusieurs images successives

Nous n'avons pas de réalité terrain pour cette application-ci. Néanmoins, nous pouvons quand même montrer la cohérence des résultats obtenus lors de la détection d'un même panneau sur des images successives. Il ne s'agit pas ici d'un suivi (bien que cela aurait pu être fait). Sur chacune des images successives, l'algorithme est exécuté de manière indépendante (donc sans prendre en compte le lien entre les images). Les zones de focalisation initiales représentent pratiquement toute l'image.

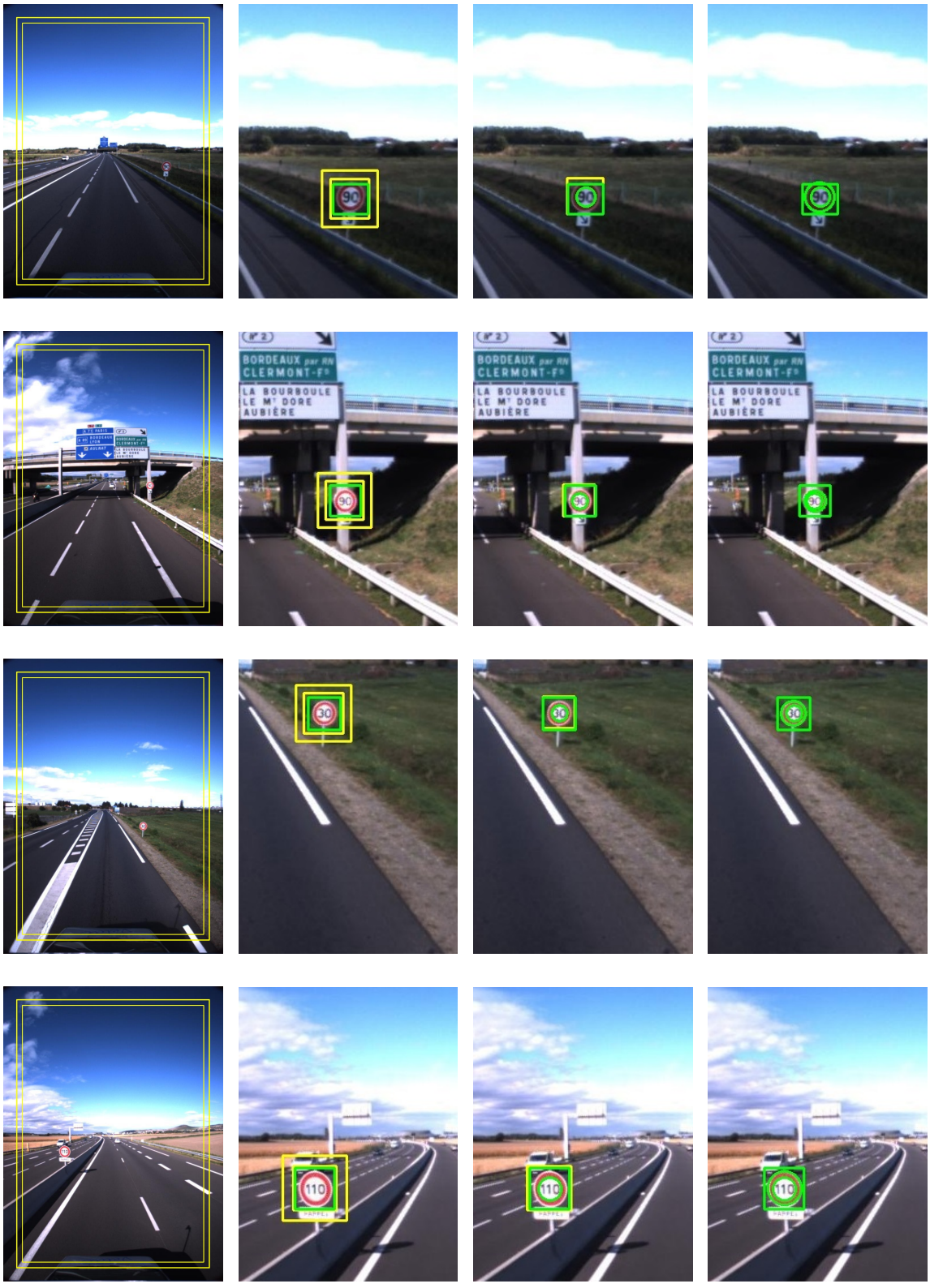


FIGURE IV.21 – Exemple de zones de focalisation. La première colonne correspond au début de l'algorithme. Les zones de focalisation sont dessinées en jaune. La zone la plus petite est celle pour la partie $petitCercle$, la zone légèrement plus grande correspond à la partie $grandCercle$, enfin la zone de focalisation pour le $blobRouge$ est toute l'image. La première colonne est la seule pour laquelle l'image complète est représentée. Pour les trois autres colonnes, seule une région autour du panneau est affichée afin de permettre une meilleure visualisation. Sur la deuxième colonne, en vert, la ***zoneRouge*** correspondant au *blobRouge* après détection est montrée. Les zones de détection pour les deux autres parties se sont réduites. La troisième colonne montre le résultat après la détection des deux premières parties (les primitives associées sont toujours dessinées en vert) et en jaune, la zone de focalisation pour la partie restante. Enfin, la dernière colonne montre le résultat final lorsque toutes les parties ont été trouvées.

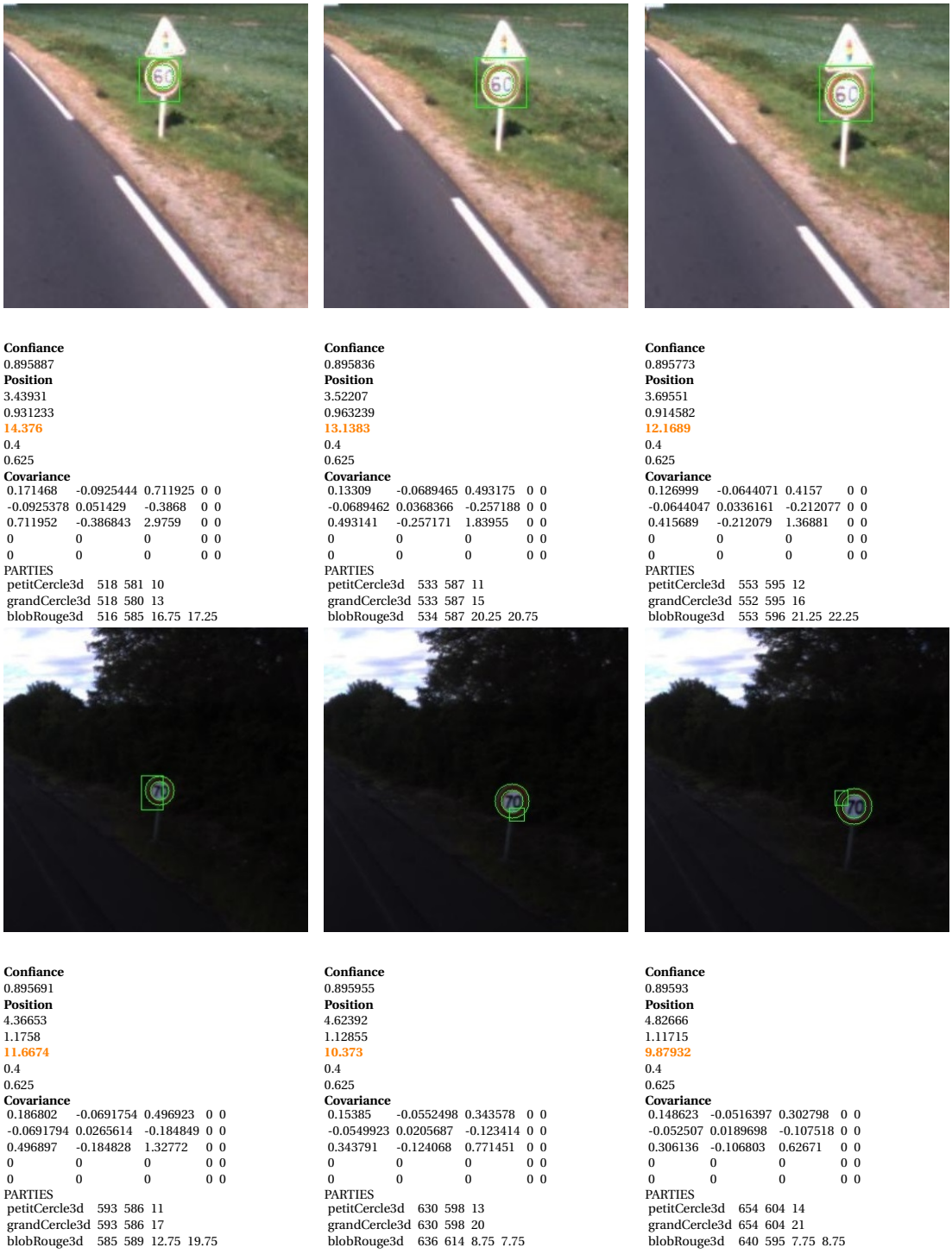


FIGURE IV.22 – Quelques exemples de résultats pour des images successives. Pour chaque série d'images, la même région contenant les panneaux est montrée pour une meilleure visualisation. En dessous de chaque image, l'estimation de l'état est présentée. Le paramètre Z, correspondant à la distance entre la caméra et le panneau, est affiché en orange. Pour la seconde série d'images, malgré la faible visibilité, l'AF parvient bien à trouver le panneau.

La figure IV.22 présente d'une part le résultat final visuel mais aussi l'état final calculé par notre algorithme. Pour avoir une estimation de profondeur, la taille du panneau et le rapport entre les rayons sont fixés selon les tailles standards. L'état final, présenté dans la figure IV.22, permet de voir que l'estimation selon X et Y varie peu au cours des images alors que la variation selon Z est plus conséquente. Cela correspond bien à ce qui est attendu sur des images successives, a priori la voiture a peu bougé latéralement et verticalement (et donc la caméra a peu bougé elle aussi) tandis qu'elle a avancé vers le panneau qui devient donc plus proche.

IV.3.3 Conclusion

Nous avons montré ici l'application de notre algorithme pour la reconnaissance de panneaux de limitation de vitesse. Cette application nous a permis d'illustrer la focalisation spatiale et de voir la cohérence lors de la mise à jour de l'état lorsque l'algorithme est appliqué sur des images successives et ce même sans suivi. Nous ne prétendons pas ici être à l'égal de l'état de l'art en la matière, d'autant plus que notre modélisation avec des cercles est assez simpliste. Néanmoins, l'AF peut se prêter à ce type d'application.

IV.4 Reconnaissance de bords de route

IV.4.1 Principe

Nous allons montrer ici l'application de notre algorithme pour la détection de bords de route. Le modèle de l'objet est construit en s'inspirant des travaux de [Aufrere et al., 2001]. Nous en présenterons brièvement le principe ici mais pour plus de détails, il faudra se référer à [Aufrere, 2001].

L'idée générale est de dire qu'une route ne peut pas avoir une forme quelconque dans une image. Il est donc possible de réaliser un apprentissage statistique pour apprendre les positions admissibles pour les bords de route. Cet apprentissage est réalisé en amont du processus général. Une fois cette étape d'apprentissage terminée, au lieu de chercher une courbe pour chacun des bords, l'image est découpée en bandelettes (voir la figure IV.23) et dans chacune des zones ainsi définie la courbe est localement approximée par un segment de droite. Trouver un bord de route revient alors à trouver les abscisses de ces segments (les ordonnées Vi étant fixées par le découpage) : c'est à dire les Ugi pour le bord gauche et les Udi pour le bord droit.

Notre algorithme devrait alors être capable de sélectionner automatiquement, grâce au critère de sélection (voir la section III.3), les zones de recherche les plus pertinentes et d'alterner les détections liées au bord droit et au bord gauche pour estimer la position des bords de la route. Le processus de remise en cause présenté au § III.6.3 permet d'éviter la confusion entre le bord droit et le bord gauche. En effet, si une primitive correspondant à une partie du bord droit est associée à une partie liée au bord gauche, les détections des parties du bord gauche échoueront, ce qui déclenchera le processus de remise en cause et permettra de faire la bonne association.

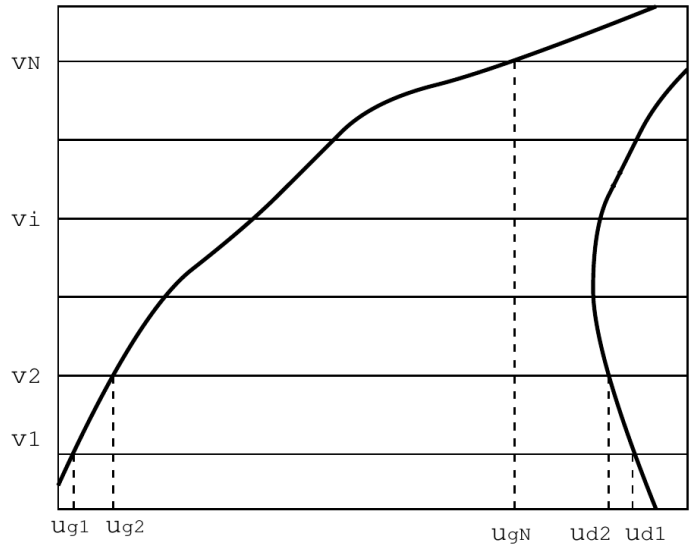


FIGURE IV.23 – Discrétisation des courbes représentatives des bords gauche et droit d'une route dans l'image issue d'une caméra embarquée dans un véhicule.

IV.4.2 Modélisation

IV.4.2.1 Modélisation de la route

Le modèle de la route est de type arc de cercle. Les paramètres utilisés pour représenter les bords de route seront les suivants :

— $x0_d$: position latérale du véhicule par rapport au bord droit de la route.

— $x0_g$: position latérale du véhicule par rapport au bord gauche de la route.

— C : courbure de la route dans le monde (positive quand la route tourne vers la droite).

— ψ : angle de cap du véhicule (positif lorsque le véhicule va vers le bord droit de la route).

— Φ : angle de tangage de la caméra (positif lorsqu'elle est inclinée vers le bas).

Le repère utilisé est montré dans la figure IV.24. Si L est la largeur de la route alors L = $x0_d - x0_g$.

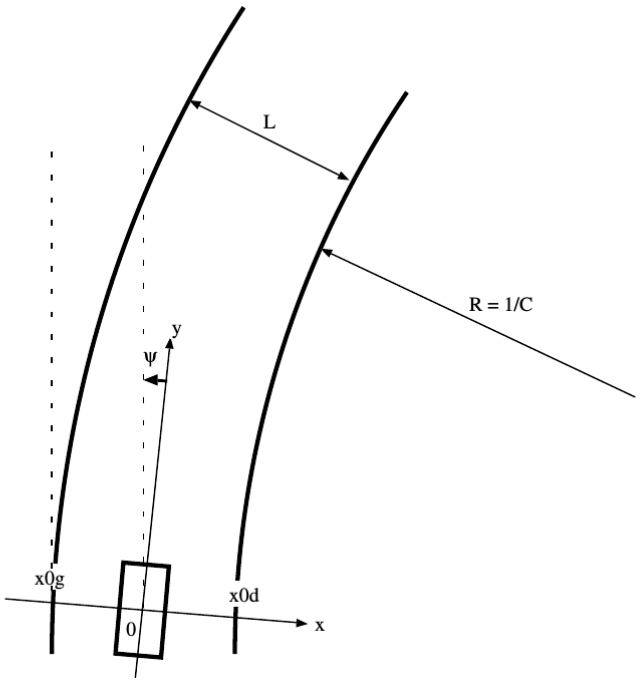


FIGURE IV.24 – Repère utilisé pour le paramétrage des bords de route.

En fonction des paramètres $x0_d$, $x0_g$, C, Φ et ψ, un ensemble de points (Udi,Vi) et (Ugi,Vi) obéissant à une loi hyperbolique est construit à l'aide des relations suivantes :

$$\begin{aligned} \mathrm{U}di &= e_u\left(\frac{e_v z_0}{2(\mathrm{V}i - e_v\Phi)}\mathrm{C} + \frac{\mathrm{V}i - e_v\Phi}{e_v z_0}x0_d + \psi\right) \\ \mathrm{U}gi &= e_u\left(\frac{e_v z_0}{2(\mathrm{V}i - e_v\Phi)}\mathrm{C} + \frac{\mathrm{V}i - e_v\Phi}{e_v z_0}x0_g + \psi\right) \end{aligned} \tag{IV.1}$$

avec :

— $e_u = \dfrac{f}{d_u}$ et $e_v = \dfrac{f}{d_v}$ où

  — f : focale de la caméra

— d_u, d_v : largeur et hauteur d'un pixel

— $e_v \psi$: hauteur de la ligne d'horizon dans l'image.

— z_0 : hauteur de la caméra

e_u, e_v et z_0 sont supposés connus.

IV.4.2.2 Modèle de l'objet "route"

Nous allons utiliser ces points pour créer le vecteur de caractéristiques de l'objet afin d'améliorer la reconnaissance :

$$\underline{\mathrm{X}} = \begin{pmatrix} \mathrm{U}d1 & \mathrm{U}g1 & \mathrm{U}d2 & \mathrm{U}g2 & \dots & \mathrm{U}d\mathrm{N} & \mathrm{U}g\mathrm{N} \end{pmatrix}^{\mathrm{T}}$$

L'apprentissage réalisé en amont, nous a permis d'obtenir :

$$\underline{\mathrm{X}}_{0|0}^{appris} = \begin{pmatrix} x0_d & x0_g & \mathrm{C} & \Phi & \psi \end{pmatrix}^{\mathrm{T}}$$

Nous disposons donc également de la matrice de covariance $\mathbf{C}_{0|0}^{appris}$ associée. Le vecteur de caractéristiques $\underline{\mathrm{X}}_{0|0}$ est obtenu grâce à la relation (IV.1). Pour obtenir la matrice de covariance initiale $\mathbf{C}_{0|0}^{appris}$ nous utilisons l'approximation suivante :

$$\mathbf{C}_{0|0} = \mathbf{H}_{appris} \mathbf{C}_{0|0}^{appris} \mathbf{H}_{appris}^{\mathrm{T}}$$

où

— $\mathbf{H}_{appris}$ est la jacobienne de $\mathbf{h}_{appris}$ avec

  — $\mathbf{h}_{appris}$ est la fonction permettant de lier le vecteur de caractéristiques initial appris $\underline{\mathrm{X}}_{0|0}^{appris}$ au vecteur de caractéristique initial souhaité $\underline{\mathrm{X}}_{0|0}$.

$$\underline{\mathrm{X}}_{0|0} = \mathbf{h}_{appris}(\underline{\mathrm{X}}_{0|0}^{appris}) \tag{IV.2}$$

Nous avons choisi d'avoir une modélisation de l'objet sous la forme d'une liste de points ($\underline{\mathrm{X}}$) plutôt que sous la forme de la liste des paramètres caractérisant la route (comme $\underline{\mathrm{X}}_{0|0}^{appris}$) car d'une part cela évite de faire des allers-retours entre la 3D et la 2D mais aussi cela permet d'avoir une mise à jour de Kalman linéaire.

IV.4.2.3 Modélisation d'une partie

Si V1,...,VN est l'ensemble des ordonnées utilisées pour le découpage de l'image en bandes alors $2(\mathrm{N}-1)$ parties seront définies : la moitié qui correspond au bord gauche et l'autre moitié au bord droit.

Pour la $i$ème partie du bord gauche, le vecteur de caractéristiques est :

$$\underline{\mathrm{Y}}^{bord\mathrm{Gauche}^i} = \begin{pmatrix} \mathrm{U}gi \\ \mathrm{U}g(i+1) \end{pmatrix}$$

et pour la $i$ème partie du bord droit :

$$\underline{\mathrm{Y}}^{bord\mathrm{Droit}^i} = \begin{pmatrix} \mathrm{U}di \\ \mathrm{U}d(i+1) \end{pmatrix}$$

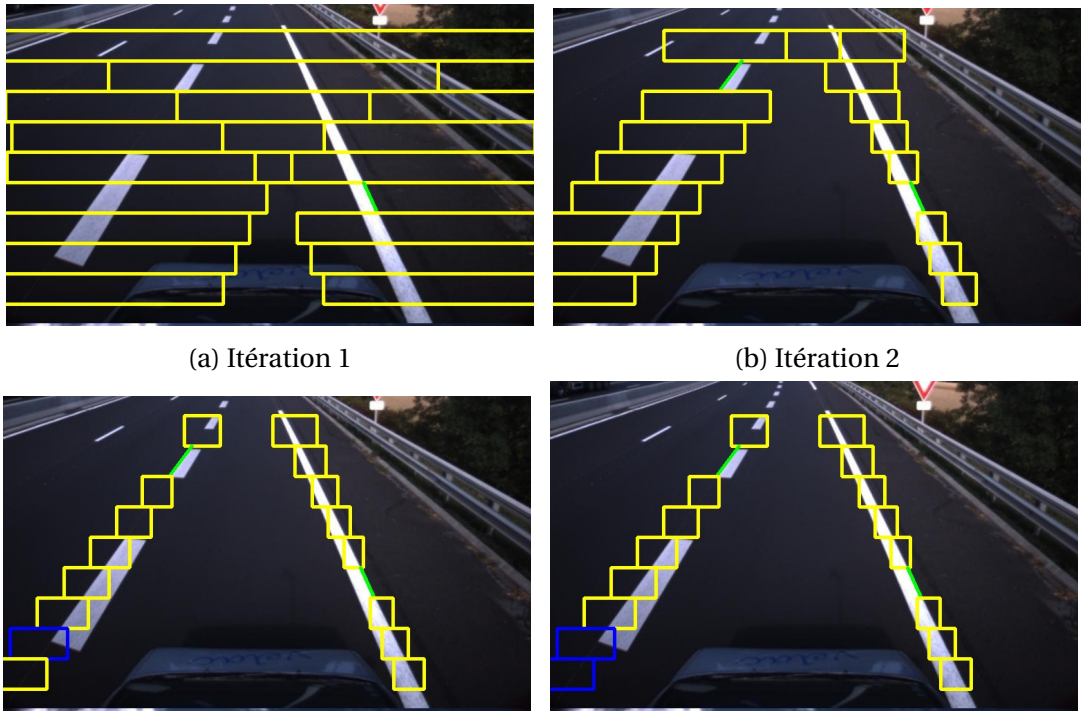
Les primitives associées sont toutes du même type et correspondent à des segments de droite. Nous disposons d'un unique détecteur de segments. Le choix du triplet se résume donc au choix de la partie.

IV.4.3 Quelques Résultats

Nous allons illustrer le principe de focalisation une nouvelle fois. Puis, nous montrerons sur un exemple l'ordre de sélection des parties constituant le bord droit et le bord gauche. Sur le même exemple, l'évolution de la confiance au fur et à mesure des tentatives de détection sera présentée.

IV.4.3.1 Focalisation

La figure IV.25 illustre l'évolution des zones de recherche au fur et à mesure que différents segments sont trouvés. Comme dans les autres exemples, les zones se rétrécissent autour des lignes cherchées au fur et à mesure que les détections progressent.



(a) Itération 1 (b) Itération 2

(c) Itération 3 (d) Itération 4

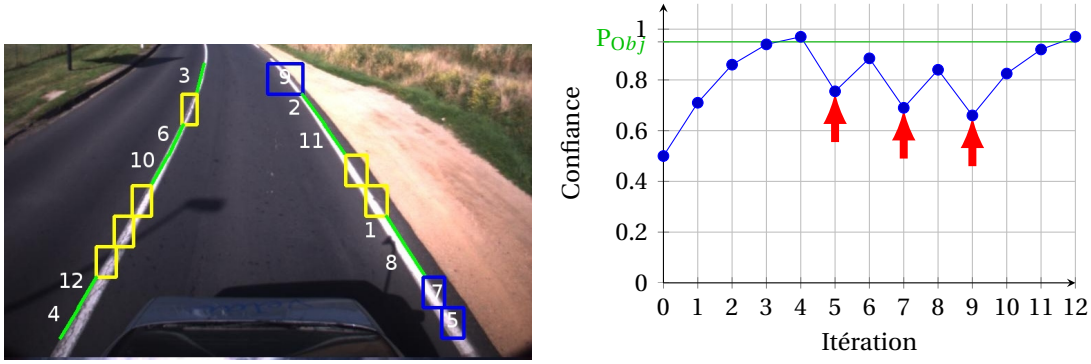
FIGURE IV.25 – Exemple de focalisation pour les zones des bords gauche et droit. En vert, les segments trouvés, les zones bleues montrent les endroits où la détection a échoué. Les zones jaunes sont les zones où aucune détection n'a encore été lancée. La détection à l'itération 1 réussit ce qui entraîne la réduction des zones de focalisation dans la sous image IV.25b. La détection réussit également lors de l'itération 2, les zones de recherche sont donc réduites dans la sous image IV.25c. Par contre, la détection tentée à l'itération 3 échoue et en conséquence, les zones de focalisation des sous images IV.25c et IV.25d sont les mêmes.

IV.4.3.2 Évolution de la probabilité sur un exemple

Sur l'image de la figure IV.26, les différents tronçons sont numérotés selon l'ordre de détection. La première partie choisie est une partie du bord droit (légèrement en dessous du milieu de la sous image IV.26a). Sa détection est une réussite, c'est ce qui est représenté par le 1 à côté du segment vert. La deuxième partie est encore une partie du bord droit mais située vers le haut de la sous image IV.26a, ce qui est représenté par le 2 accolé à un segment vert, probablement pour confirmer que les segments détectés appartiennent bien à un bord. La troisième partie choisie est une partie du bord gauche située vers le haut de

la sous image IV.26a, sans doute afin de préciser la largeur de la route. La quatrième partie choisie est une partie du bord gauche située en bas de la sous image IV.26a et ainsi de suite. Les parties choisies alternent donc entre le bord gauche et le bord droit et sont réparties tout le long des bords afin d'améliorer au mieux la précision.

Les détections des itérations 5, 7 et 9 échouent, elles sont représentées par les rectangles bleus, le numéro de l'itération correspondante est inscrit à l'intérieur du rectangle. Ces échecs correspondent bien à une diminution de la confiance dans le graphe IV.26b. Nous notons d'ailleurs qu'après la quatrième itération, la confiance était supérieure à l'objectif de confiance (représenté par la ligne verte dans la figure IV.26b) mais l'algorithme ne s'arrête pas car l'objectif de précision n'est pas encore atteint. L'algorithme se termine à la douzième itération. Il reste d'ailleurs encore des parties à détecter, leurs zones de recherche sont dessinées en jaune dans l'image IV.26a. En effet, les objectifs de précision et de confiance sont atteints. Des objectifs plus ambitieux entraîneraient la détection de plus nombreuses parties



(a) Ordre de détection des différentes parties

(b) Évolution de la confiance en fonction des itérations

FIGURE IV.26 – Ordre de détection des parties et évolution de la confiance en fonction des itérations. Dans l'image IV.26a, les numéros indiquent l'itération à laquelle une détection est lancée. Lorsque le numéro est à côté d'un segment vert, la détection de cette itération a réussi et le segment vert représente la primitive trouvée. Quand le numéro est dans un rectangle bleu, c'est que la détection a échoué. Les rectangles jaunes représentent les zones de recherche où aucune détection n'a été tentée. Dans la figure IV.26b, les points marqués par des flèches rouges correspondent aux itérations où les détections ont échoué. La confiance augmente lorsque les détections réussissent et diminue lorsqu'elles échouent. La ligne verte indique l'objectif de confiance souhaité.

IV.4.4 Conclusion

Nous venons de montrer une autre application possible de notre algorithme de perception pour la reconnaissance de bords de route. L'algorithme est capable de se focaliser de manière cohérente autour des différentes parties. La probabilité d'existence des parties lui permet de gérer des routes avec des bords discontinus (et qui donc localement peuvent être absents). Nous avons vu l'évolution de la confiance lors d'un exemple de détection avec une confiance qui diminue effectivement en cas de non détection mais qui n'entraîne pas de remise en cause immédiate et permet de trouver les autres parties.

IV.5 Estimation fine de la pose d'un véhicule

IV.5.1 Introduction

L'estimation fine de la pose d'un objet est un problème important pour bien comprendre et interagir avec l'environnement : trouver simplement l'objet dans une image est souvent un premier pas mais qui ne fournit pas nécessairement assez d'informations (il s'agit le plus souvent d'une boîte englobante de l'objet) pour de futures interactions. Les auteurs de [Lim et al., 2014] donnent l'exemple d'un robot autonome humanoïde qui veut s'asseoir sur une chaise : avoir une boîte englobante pour la chaise n'est pas très utile au contraire d'une estimation précise de la pose de la chaise.

Détecter des objets 3D dans des images et estimer leur pose était un sujet de recherche populaire dans les premières années de la vision par ordinateur [Lowe, 1991, Mundy, 2006] qui a regagné en popularité ces dernières années [Lim et al., 2014]. Avec le succès de la vision par ordinateur pour la détection d'objets et la compréhension de scènes [Felzenszwalb et al., 2010, Girshick et al., 2014], l'estimation fine de pose devient de plus en plus accessible.

Dans l'application que nous allons présenter ci-après, nous nous intéresserons au problème particulier de l'estimation fine de pose pour les véhicules. En effet, la connaissance précise de la pose d'un véhicule a de multiples applications en particulier dans les domaines de la conduite autonome, des systèmes de parking intelligent ou de manière plus générale pour les systèmes d'aide à la conduite.

IV.5.2 Hypothèses de travail

Nous partons du principe qu'un modèle 3D du véhicule recherché est disponible. Il faut noter qu'il est de plus en plus aisé de trouver des modèles 3D pour les objets manufacturés : en effet de tels modèles servent à la construction des objets (par exemple en permettant d'étudier les forces exercées sur les différentes pièces) mais aussi pour des applications de réalité virtuelle qu'elles soient à but ludiques (dans des jeux vidéos par exemple) ou commerciaux et surtout pour la publicité et la communication.

Comme la plupart des applications d'estimation fine de pose [Hsiao et al., 2014, Prokaj and Medioni, 2009, Zia et al., 2011], nous faisons l'hypothèse que nous disposons d'un détecteur capable de donner une estimation grossière de la pose du véhicule : ce détecteur est supposé renvoyer une boîte englobante de la voiture ainsi qu'un angle grossier sous lequel le véhicule est vu (dans notre cas, il renverra une valeur par tranche de 45°).

En connaissant le modèle 3D de l'objet, si nous sommes capables de trouver les correspondances entre certains points 3D appartenant au modèle avec des points 2D dans notre image (qui correspondront donc à nos parties) alors le problème de l'estimation fine de pose peut être transformé en un problème d'estimation de la pose de la caméra. Au lieu d'avoir une caméra fixe et de chercher la position dans le monde de la voiture par rapport à la caméra, la position de la voiture est supposée connue et c'est la caméra qui se déplace autour de la voiture pour la voir comme dans l'image. Si, de plus, il est admis que les paramètres intrinsèques sont connus (ou qu'il est possible d'en faire une estimation

valable), le problème devient un problème de Perspective-n-Point **Pnp** pour lequel de nombreuses solutions existent [Abdel-Aziz and Karara, 2015, Lepetit et al., 2009]. Toutes ces solutions nécessitent d'avoir les correspondances correctes entre les points 3D et les points 2D.

IV.5.3 Modélisation

Notre objet sera donc modélisé par les 6 paramètres (3 pour la translation et 3 pour la rotation) extrinsèques de la caméra :

$$\underline{\mathrm{X}}_{3\mathrm{D}} = \begin{pmatrix} \theta_x & \theta_y & \theta_z & t_x & t_y & t_z \end{pmatrix}^{\mathrm{T}}$$

Si notre objet 3D est défini par M points 3D $\mathrm{P}_i = (\mathrm{X}_i, \mathrm{Y}_i, \mathrm{Z}_i)^{\mathrm{T}}$ pour $i = 1, ..., \mathrm{M}$, nous pouvons alors définir M parties correspondant à la projection dans l'image de ces M points. La figure IV.27 montre les parties qui seront utilisées pour l'estimation de la pose du véhicule.

$$\underline{\mathrm{Y}}^i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$$

Pour le calcul de la projection, plus de détails ont déjà été donnés dans IV.3.1.3.a . M_{calib} est exprimée en fonction de M_{int}, la matrice intrinsèque de calibration de la caméra supposée connue et M_{ext}, la matrice extrinsèque de calibration de la caméra :

$$\mathrm{M}_{calib} = \mathrm{M}_{int}\,\mathrm{M}_{ext}$$

M_{ext} peut être calculée à partir des valeurs du vecteur de caractéristiques $\underline{\mathrm{X}}_{3\mathrm{D}}$:

$$\mathrm{M}_{ext} = \begin{pmatrix} & & & t_x \\ & \mathrm{R} & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

avec :

$$\begin{aligned} \mathrm{R} &= \mathrm{R}_x \mathrm{R}_y \mathrm{R}_z \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{pmatrix} \begin{pmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{pmatrix} \begin{pmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

En fait, ce n'est pas directement le vecteur de caractéristiques de l'objet qui va servir pour l'estimation mais la liste des points 2D représentant les parties qui le composent. Dans cette application, les résultats sont meilleurs en procédant ainsi plutôt qu'en conservant le vecteur de caractéristiques lié à la 3D. En effet, cela évite des problèmes lors de la mise à jour de l'estimation avec le filtre de Kalman liés aux non linéarités du modèle.

$$\underline{\mathrm{X}}_{2\mathrm{D}} = \begin{pmatrix} u_1 & v_1 & \dots & u_{\mathrm{M}} & v_{\mathrm{M}} \end{pmatrix}^{\mathrm{T}}$$

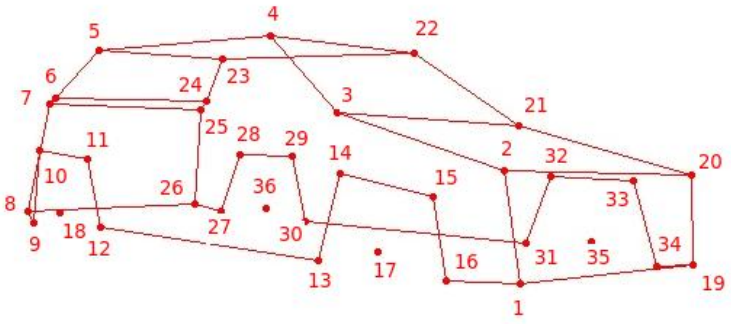


FIGURE IV.27 – Modèle de décomposition d'une voiture en parties.

IV.5.4 Initialisation

Le vecteur représentant les paramètres extrinsèques de la caméra sera initialisé à l'aide de la détection du détecteur grossier. La transformation en vecteur de caractéristiques contenant les points 2D recherchés est ensuite réalisée. La confiance initiale $P(\mathbf{E}_{0|0})$ vaut 1 : nous faisons l'hypothèse que le détecteur grossier a renvoyé une zone de l'image où se trouve réellement la voiture. Pour les angles, les valeurs suivantes seront utilisées : $\theta_x = 0$, $\theta_z = 0$ et $\theta_z = angleGrossier$ où $angleGrossier$ est l'angle estimé par le détecteur grossier.

Pour initialiser la translation, les paramètres de translation t_x, t_y, t_z sont déterminés afin que le centre de la voiture soit au centre de l'image et que la voiture estimée occupe presque toute l'image. Plusieurs valeurs de zooms pourront être testées.

La figure IV.29 donne quelques exemples d'initialisation.

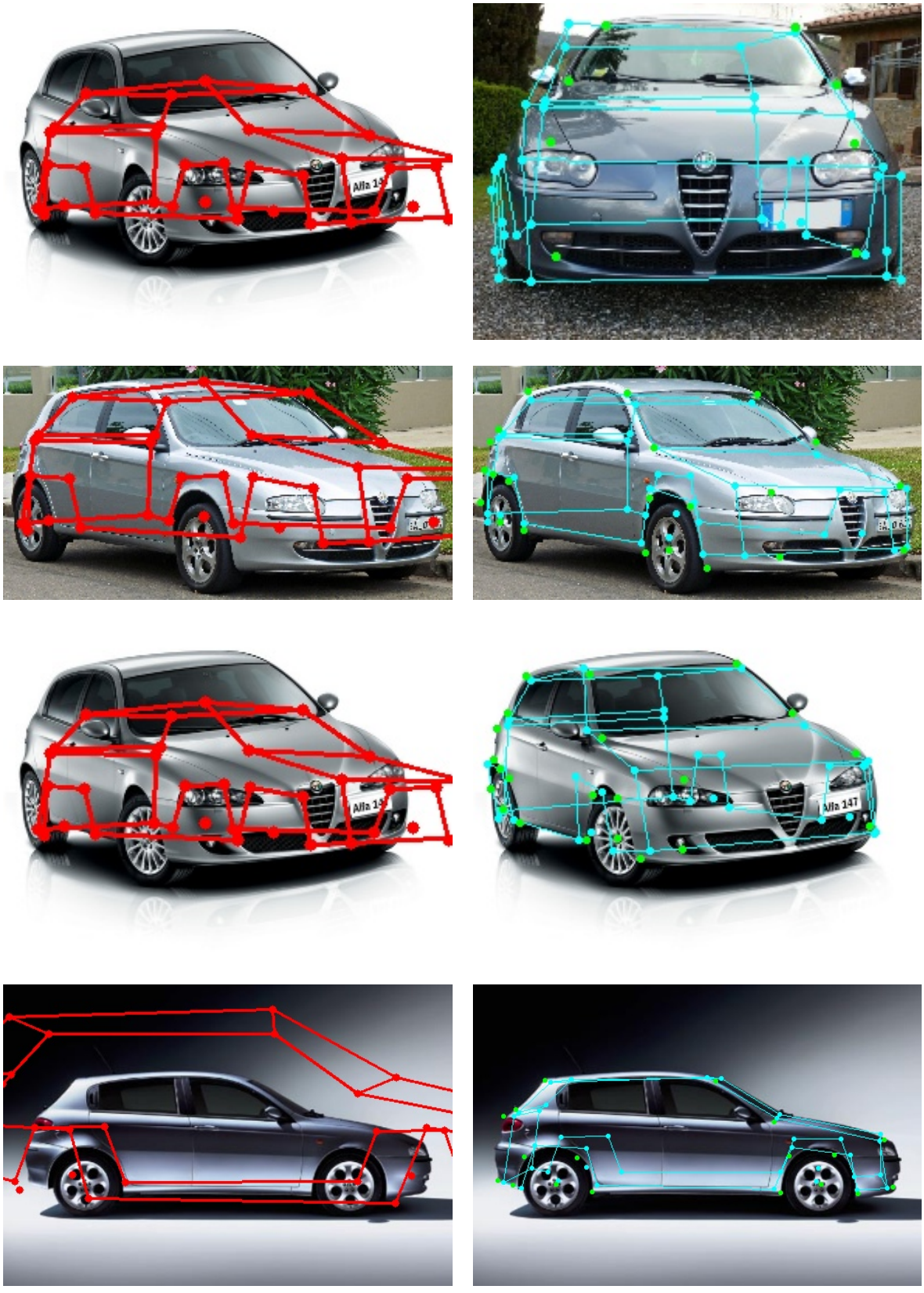
IV.5.5 Détecteurs

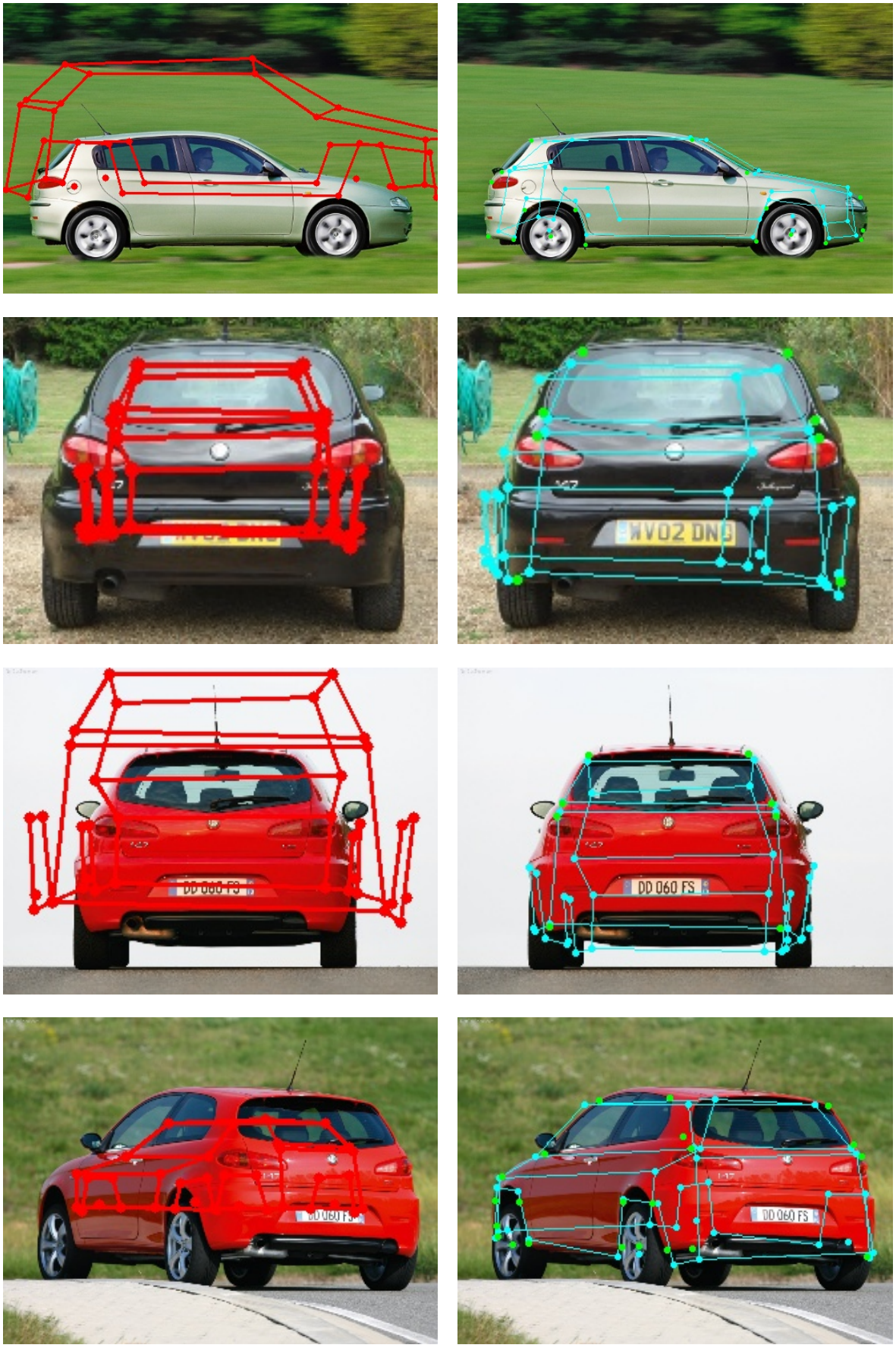
Pour les détecteurs, nous utilisons ceux des auteurs de [Chabot et al., 2015] qui utilisent des réseaux de neurones convolutionnels (Convolutionnal Neural Network **CNN**). Ces détecteurs ont été entraînés pour chaque partie. La figure IV.28 montre quelques résultats de ces détecteurs. Certains détecteurs peuvent avoir du mal à renvoyer une primitive compatible tandis que d'autres à l'inverse auront tendance à renvoyer trop de primitives. Certains détecteurs ont également de bons résultats. Il est expérimentalement possible de qualifier la qualité moyenne de chaque détecteur (en terme de précision et de probabilité de détection). C'est ce que nous avons fait afin de déterminer les taux de faux positifs β et faux négatifs α des détecteurs ainsi que leur précision.

IV.5.6 Résultats

Nous allons maintenant présenter les résultats pour un unique modèle de voiture mais vu sous différents angles et dans des images variées. Nous présentons dans la figure IV.29 les résultats obtenus. La première colonne correspond à l'initialisation utilisée (selon le bon ou le mauvais hasard, elle est plus ou moins proche des données), la seconde colonne montre le résultat final de l'algorithme : en cyan le squelette estimé de la voiture et en vert les points annotés comme étant réellement la partie cherchée (vérité terrain).

FIGURE IV.28 – Exemple de résultats pour les détecteurs de parties de voiture. Chaque ligne correspond à un détecteur différent. La bonne position pour la partie est encadrée en bleu. Les ronds jaunes représentent les primitives renvoyées par le détecteur concerné.




WV02 DNG
DD 060 FS

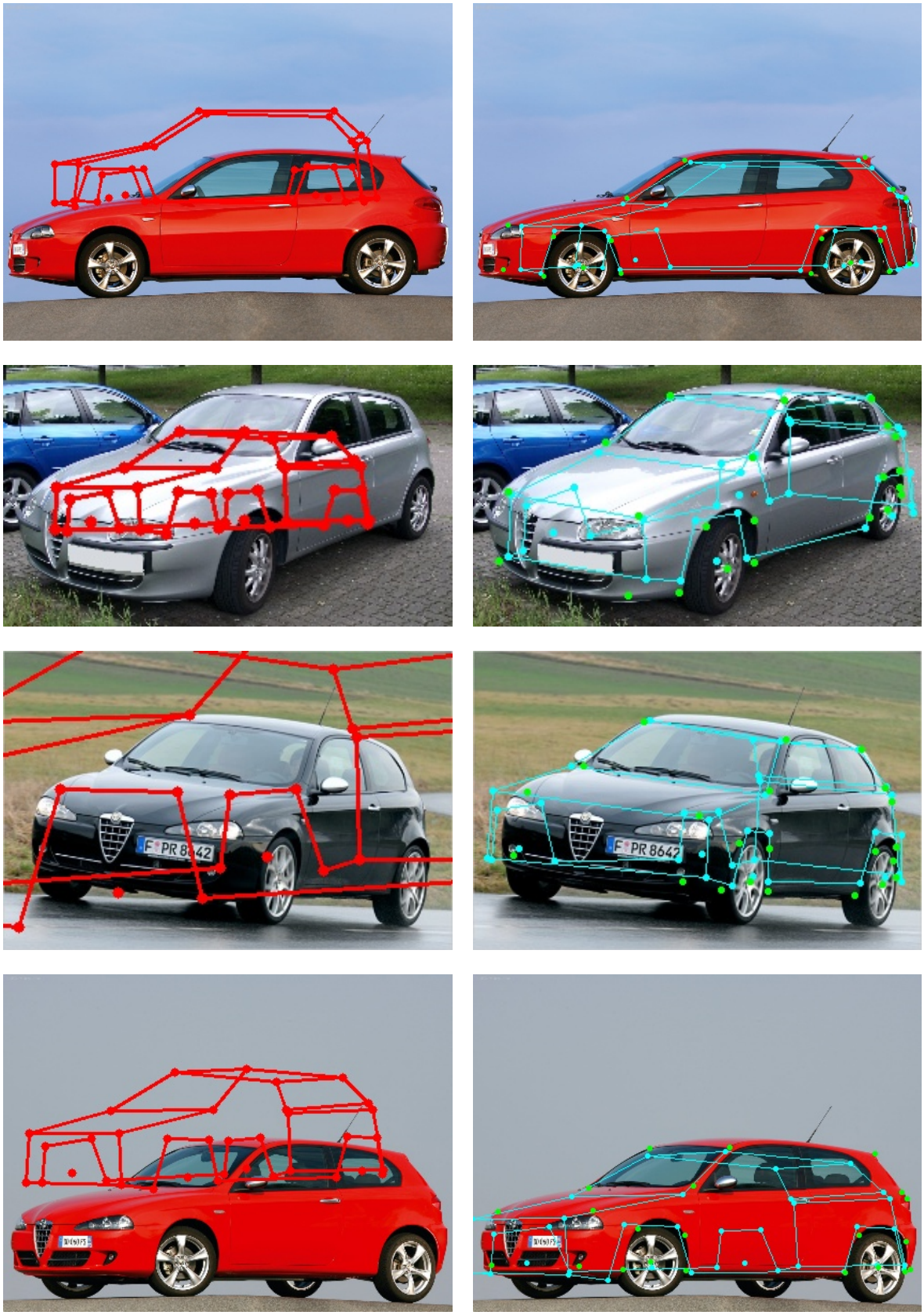


FIGURE IV.29 – Quelques exemples de résultats pour l'estimation fine de pose d'un véhicule. La première colonne montre en rouge l'initialisation utilisée pour parvenir au résultat. La deuxième colonne montre l'image résultante finale avec en cyan la pose estimée et en vert les points réels.

IV.5.7 Commentaires et perspectives

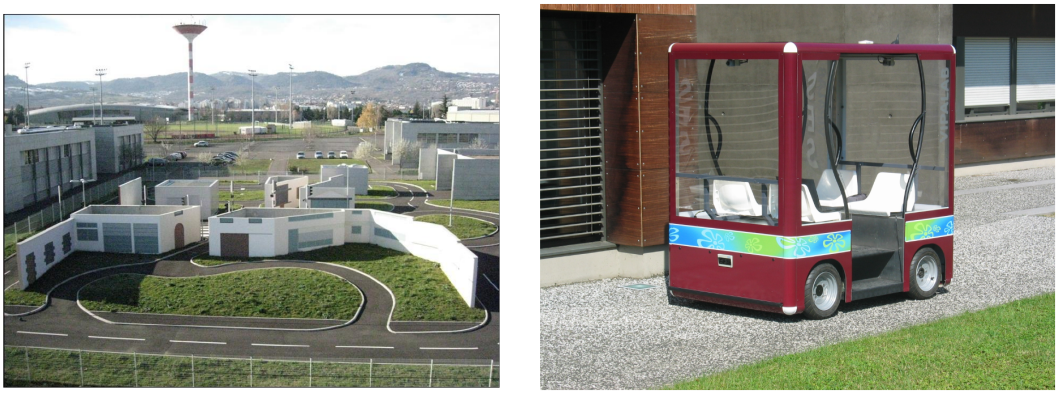
Nous avons montré une application de notre algorithme pour l'estimation fine de pose. Les résultats obtenus sont encourageants et mériteraient d'être poursuivis : qualitativement pour un modèle de voiture, l'estimation semble correcte. Il reste quelques points à améliorer : bien qu'une initialisation automatique ait été proposée, il est utile de faire varier la distance utilisée afin de pouvoir obtenir une meilleure estimation. Dans la figure IV.29, la voiture initiale apparaît plus ou moins lointaine selon les images. Il serait également judicieux de rendre l'algorithme plus robuste à un changement d'initialisation. Les résultats peuvent grandement varier quand l'initialisation est modifiée. Par contre, l'algorithme est bel et bien capable de gérer les absences de détection ou de choisir un candidat acceptable quand il y a de multiples primitives disponibles.

IV.6 Localisation d'un véhicule à l'aide d'une carte

Jusqu'ici nous avons présenté des applications qui étaient plutôt orientées reconnaissance d'objets mais notre algorithme peut aussi tout à fait s'appliquer à des problèmes de localisation de robot mobile. Ici nous allons montrer quelques résultats de [Aynaud, 2015] pour des problématiques de localisation sachant que les fondements algorithmiques sont identiques.

IV.6.1 Environnement de tests

Les tests présentés ont été réalisés en environnement réel sur la plate-forme d'expérimentation Pavin (voir la figure IV.30a). Ce site comprend deux zones : la première reproduit un environnement urbain avec des rues, des rond points, des feux tricolores fonctionnels et la deuxième correspond à une zone rurale. Au total, les deux zones couvrent environ 5000 m^2. Le site Pavin est couvert par une station de base DGPS ce qui permet d'avoir une localisation RTK précise (et qui pourra ainsi servir pour la suite de vérité terrain).



(a) site expérimental Pavin (b) véhicule VipaLab

FIGURE IV.30 – environnement et véhicule utilisés pour les tests

Les tests sont effectués sur des véhicules électriques nommés VipaLab (voir la figure IV.30b) qui peuvent être équipés de différents capteurs :

- odomètre : donne le mouvement relatif des roues.
- capteur d'angle au volant.
- télémètres Laser, SICK LMS151
  - portée : jusqu'à 30 mètres
  - angle de vue : de -45 à 225°
  - résolution : 0.5°
- GPS standard : précision de l'ordre de 5 mètres
- GPS RTK, ProFlex 800, avec une précision centimétrique. Il est utilisé pour avoir une vérité terrain afin de quantifier la qualité des résultats (il ne sera donc jamais utilisé directement par l'algorithme mais servira uniquement de base de comparaison).

IV.6.2 Modélisation

L'objet à estimer sera ici la position du véhicule dans le plan ainsi que le cap du véhicule.

$$\underline{X} = \begin{pmatrix} X & Y & \Phi \end{pmatrix}^T$$

Le principe de la localisation est le suivant : nous disposons d'une carte précise de l'environnement statique. À chaque instant, le système recherche l'**amer** et le capteur le plus à même de réduire l'incertitude et d'augmenter la confiance dans l'estimation de position.

Les parties correspondront à des amers de la carte. Ainsi, les murs, les trottoirs et autres amers potentiellement reconnaissables ont été cartographiés et leur position réelle dans le monde est supposée connue. Détecter un amer permettra alors de préciser la position du véhicule. La détection d'un mur pourra par exemple correspondre à la détection d'une ligne dans une image télémètre.

Le véhicule est initialisé avec une position correcte à 5 mètres près et le cap initial sera donné à 10° près.

IV.6.3 Localisation à l'estime

La localisation à l'estime consiste à estimer la position du véhicule en se basant uniquement sur les mesures des déplacements élémentaires effectués depuis la dernière estimation. Les capteurs utilisés pour cette localisation à l'estime sont le plus souvent des capteurs proprioceptifs. Ici, il s'agira d'un odomètre et d'un capteur d'angle au volant. Le modèle d'évolution du véhicule est supposé assimilé au modèle bicyclette, ce qui est courant dans ce genre d'application. Les véhicules VipaLab circulent à une vitesse comprise entre 5 et 10 km/h.

Pour cette première localisation, notre algorithme n'intervient donc pas : il n'y a pas de détection d'amers extérieurs au véhicule donc pas de sélection de triplets etc. Cette étape permet de voir les améliorations qui seront apportées par le jeu des détections. De plus, nous utiliserons, pour les tests suivants, ce système comme phase de prédiction dans le filtre de Kalman utilisé pour la mise à jour (voir le § III.6.2).

La figure IV.31 montre les résultats obtenus. La forme globale de la trajectoire est bien la même que celle de la trajectoire de référence mais sans surprise ce système accumule les erreurs au cours du temps ce qui génère une forte dérive. Il est possible de remarquer qu'il n'y a pas de trajectoire de référence lorsque le véhicule passe dans le garage : cela est dû à la non réception des données GPS à cet endroit là.

IV.6.4 Localisation avec un capteur d'angle absolu

Un capteur permettant une estimation de l'angle d'orientation du véhicule par rapport au nord est ajouté et va nous permettre d'avoir notre premier triplet perceptif. Cette fonction pourrait être réalisée avec un magnétomètre mais pour cette application, il a été simulé grâce à l'orientation fournie dans les trames du récepteur GPS Ublox. En conséquence, ce capteur ne sera pas pertinent lorsque le véhicule circule dans les bâtiments ou lorsqu'il est à l'arrêt.

FIGURE IV.31 – Résultats pour la localisation à l'estime. La trajectoire estimée est représentée en noir. Le point de départ est représenté en rouge, le point d'arrivée estimé par le rectangle jaune, le point d'arrivée réel par la flèche magenta. La trajectoire de référence (donnée par le GPS-RTK) apparaît en cyan. Les différents éléments référencés de la carte sont représentés en bleu.

FIGURE IV.32 – Résultats pour la localisation avec un unique capteur d'angle absolu. Le point de départ est représenté en rouge, le point d'arrivée estimé par le rectangle jaune, le point d'arrivée réel par la flèche magenta. La trajectoire de référence (donnée par le GPS-RTK) apparaît en cyan. Les différents éléments référencés de la carte sont représentés en bleu.

Notre algorithme va être utilisé ici. Une initialisation cohérente est donnée à l'algorithme. Il y a un seul et unique triplet disponible : la partie correspond au véhicule, le capteur est le GPS, le détecteur est l'algorithme permettant de calculer l'angle absolu à partir des données GPS. Les valeurs estimées pour ce dernier sont utilisées pour la mise à jour. La phase de prédiction du filtre de Kalman, sera réalisée avec la localisation à l'estime vue précédemment.

La figure IV.32 montre les résultats pour la localisation. L'estimation de la trajectoire est bien meilleure mais le problème de dérive subsiste partiellement et la précision obtenue n'est pas suffisante pour envisager un guidage automatique du véhicule.

IV.6.5 Localisation avec un télémètre laser

Dans cet exemple aussi, un seul capteur est utilisé : il s'agit d'un télémètre laser situé à l'avant du véhicule. Par contre, contrairement à précédemment, plusieurs triplets différents pourront être utilisés. Ils auront tous le même capteur (le télémètre laser) et le même détecteur (un détecteur de lignes) mais des amers différents correspondant aux murs cartographiés. L'algorithme sélectionnera un mur en fonction de sa visibilité, de la précision et de l'information qu'il apportera, compte tenu notamment de l'estimation de la position du véhicule.

La figure IV.33 montre les résultats dans cette configuration. L'erreur de l'estimation est montrée dans la figure IV.34. L'erreur moyenne d'estimation sur la trajectoire est de $6\,cm$ avec un pic à $44\,cm$ au départ (le temps que le système recale convenablement son incertitude). L'erreur reste ensuite généralement en deçà de $10\,cm$ avec néanmoins des pics supérieurs à $15\,cm$. Ces pics d'erreur correspondent aux moments où le véhicule ne parvient pas à voir les amers alors qu'il devrait pouvoir le faire : la pente du terrain n'est pas prise en compte et à certains endroits elle est assez forte pour empêcher la détection par le télémètre. En conséquence, la mise à jour de position ne peut pas se faire.

IV.6.6 Localisation avec quatre télémètres laser

Notre algorithme a été conçu pour fonctionner avec plusieurs capteurs. Nous le testons ici avec quatre capteurs identiques : des télémètres laser situés aux quatre coins du véhicule. Par rapport au cas précédent, il y a donc potentiellement quatre fois plus de triplets. Il y a en effet quatre choix possibles pour le capteur, le détecteur reste le même et les parties disponibles sont également les mêmes (les murs cartographiés).

La trajectoire parcourue est représentée dans la figure IV.35. La précision de l'estimation pour cette trajectoire est montrée dans la figure IV.36. Elle est moins bonne qu'avec un seul télémètre : l'erreur moyenne est égale à $8\,cm$ alors qu'avec un seul télémètre elle était de $6\,cm$. Cette différence n'est pas due à la méthode en elle-même mais à un souci de temps de calcul. En effet, avec quatre télémètres il y a environ quatre fois plus de triplets. Donc le temps nécessaire à la phase de sélection augmente fortement, ce qui entraîne qu'un plus grand nombre de données capteur ne sont pas traitées. Par conséquent, la qualité de l'estimation diminue.

Les temps de calcul du meilleur triplet occupent environ 30% du temps de calcul. Or ils sont dus essentiellement aux calculs des probabilités dans les réseaux bayésiens qui

FIGURE IV.33 – Résultats pour la localisation avec un unique télémètre. La trajectoire estimée est représentée en noir. La position de départ, et les positions d'arrivée (estimée et réelle) sont au niveau du rectangle jaune. La trajectoire de référence (donnée par le GPS-RTK) apparaît en cyan.

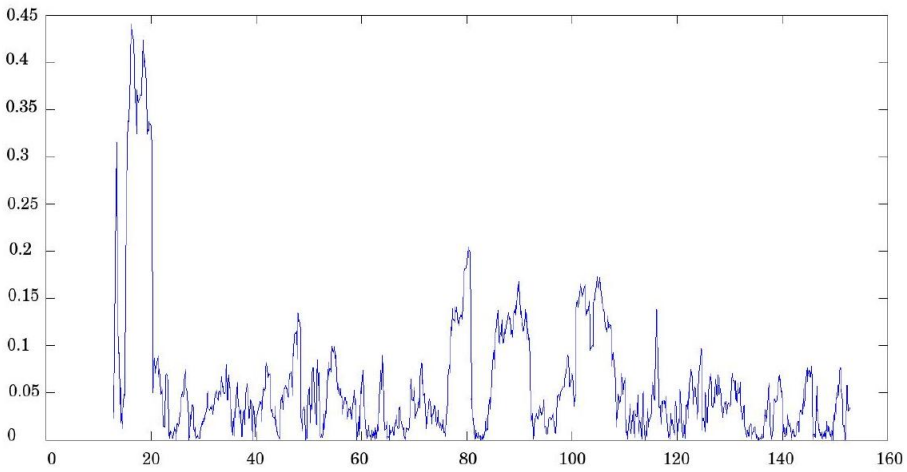


FIGURE IV.34 – Erreur de localisation (en m) au cours du temps (s) lors de l'estimation de la trajectoire avec un télémètre laser.

sont fait avec la bibliothèque Dlib [King, 2009]. Il serait donc possible de réduire ces temps en utilisant la méthode proposée en annexe C, ce qui n'a pas été fait ici. Cette méthode optimise les calculs de probabilités dans les cas de réseaux bayésiens à structure fixe et à événements binaire (ce qui correspond à notre cas).

FIGURE IV.35 – Résultats pour la localisation avec quatre télémètres laser. La trajectoire estimée est représentée en noir. La trajectoire de référence (donnée par le GPS-RTK) apparaît en cyan. Le point de départ est dessiné en rouge. Le point d'arrivée est représenté par le rectangle jaune.

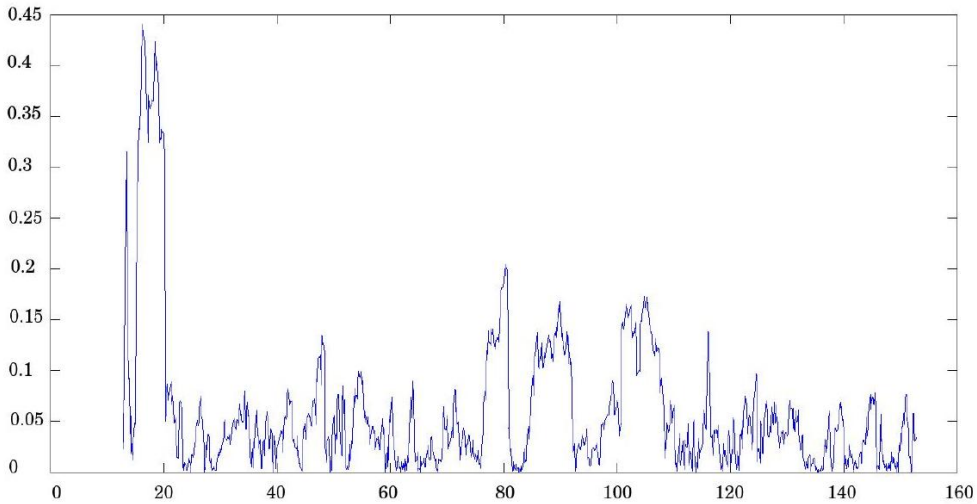


FIGURE IV.36 – Erreur de localisation (en m) au cours du temps (s) lors de l'estimation de la trajectoire avec quatre télémètres laser.

IV.6.7 Conclusion

Nous avons montré que notre algorithme pouvait parfaitement s'adapter pour une application de localisation de robot mobile. Nous avons eu l'occasion de montrer que différents capteurs pouvaient être utilisés, ensemble ou séparément, de manière naturelle par l'algorithme. Plus de résultats pour cette application de localisation sont disponibles dans [Aynaud, 2015].

IV.7 Conclusion

Les diverses expérimentation que nous avons menées montrent que l'algorithme que nous avons développé peut être utilisé pour de multiples applications dont les objectifs sont complètement différents comme la reconnaissance d'objets ou la localisation de robot mobile. Notre algorithme est capable de sélectionner à chaque instant le meilleur triplet. L'ajout d'un nouveau capteur ou d'un nouveau détecteur ne modifie aucunement la conception de l'algorithme (qui calcule seulement plus de triplets).

L'ajout d'un paramètre de confiance permet d'avoir un processus de remise en cause. Ce dernier prend effet lorsque une ou plusieurs détections échouent et que la raison la plus probable de ces échecs est une mauvaise estimation. Lorsqu'il est nécessaire, le retour en arrière permet de revenir sur une association précédente et de modifier le choix qui avait été fait afin de parvenir à une nouvelle estimation.

Quelle que soit l'application, la focalisation spatiale permet de diminuer les temps de calculs en n'exécutant les détecteurs que dans des zones pertinentes. De plus, elle permet également, avec la focalisation de caractéristiques, de limiter l'influence du bruit sur l'algorithme. Le bruit aura toujours une influence plus forte au début lorsque l'estimation est peu précise. Dans ce cas, l'élimination des outliers est plus difficile.

Nous avons également pu remarquer que les résultats étaient souvent meilleurs en transformant le vecteur de caractéristiques de l'objet lorsque celui-ci contenait des données liées à la 3D alors que les observations étaient réalisées en 2D. Il est préférable dans ces cas-ci de modifier le vecteur de caractéristiques 3D originel pour obtenir un vecteur de caractéristiques 2D. Cela évite d'une part des allers-retours entre la 3D et la 2D et permet une mise à jour de l'estimation avec le filtre de Kalman linéaire.

Chapitre V

Conclusion et Perspectives

Sommaire

V.1 Conclusion

Dans ce manuscrit, nous nous sommes intéressés à proposer un algorithme polyvalent et générique reposant sur une stratégie de perception active. Nous avons vu dans les chapitres I et II que la perception humaine était particulièrement efficace et avait déjà inspiré de nombreux algorithmes. Nous nous sommes orientés plus particulièrement vers les méthodes top-down, en s'inspirant des processus endogènes de la perception humaine, avec des modèles d'objets en parties. Nous avons cependant conservé des aspects utilisés également dans les système bottom-up et permettant d'optimiser la détection tels que le principe de la cascade de détecteurs ou encore de la focalisation spatiale. Celle-ci évite d'utiliser les détecteurs dans toute l'image, ce qui peut s'avérer potentiellement coûteux en terme de temps de calcul.

Nous avons formulé, dans le chapitre II, le problème de perception comme étant un problème d'estimation d'un vecteur de caractéristiques de l'objet. Nous avons alors proposé un état de l'objet nous permettant à chaque itération de définir l'estimation courante du vecteur de caractéristiques. Une matrice de covariance associée permet de caractériser la précision estimée. Mais de manière plus originale, nous avons également défini une confiance dans cette estimation. Notre objectif de perception devient alors un double objectif qui exige une précision et une confiance minimales.

Pour un objet, nous disposons de plusieurs parties, capteurs et détecteurs. Nous avons alors voulu sélectionner l'association de ces trois éléments, appelée triplet, la plus pertinente pour améliorer notre estimation en fonction des objectifs fixés. Nous avons donc défini un critère nous permettant de faire une sélection efficace en fonction des informations disponibles sur l'environnement, des détecteurs et de l'estimation courante.

Nous avons vu que pour ce critère nous avions besoin de prendre en compte des événements incertains comme l'incapacité potentielle d'un détecteur à détecter convenablement une partie sélectionnée. Pour permettre le calcul des probabilités associées, nous avons utilisé un réseau bayésien. Celui-ci a plusieurs avantages en plus de résoudre les problèmes de calculs de probabilité de plus en plus complexe à chaque ajout de nouvel élément : il facilite ainsi la modélisation du problème en permettant une visualisation simple des événements pris en compte. De plus, ce réseau est facilement modifiable ce qui permet éventuellement une complexification progressive.

Dans notre algorithme, ce réseau bayésien, en plus d'être utilisé pour le critère de sélection est également indispensable pour la mise à jour de la confiance. Il permet d'avoir une évolution cohérente de la confiance en fonction de l'échec ou de la réussite de la détection. Il prend également en compte le nombre de candidats trouvés ou encore la qualité du détecteur. Enfin, c'est lui qui permet de remettre en cause les mauvaises associations qui auraient pu être faites lorsqu'il y avait plusieurs choix possibles.

Nous avons également présenté le principe de focalisation à la fois en terme de focalisation spatiale mais également en terme de focalisation de caractéristiques. Cette double focalisation présente l'avantage de limiter l'influence du bruit en obligeant ainsi une grande cohérence entre les détections successives et en augmentant le rapport signal sur bruit. Du fait que les détecteurs sont utilisés dans des zones restreintes, ces focalisations permettent également de diminuer les temps de calculs.

Enfin, après avoir détaillé les différentes étapes (sélection, détection et mise à jour) qui composent notre algorithme dans le chapitre III, nous avons présenté son utilisation pour différentes applications qu'il s'agisse de reconnaissance d'objets ou bien de localisation de robot mobile. Certaines applications comme la reconnaissance de panneaux de limitation de vitesse ou de polygones à M côtés permettent surtout de se familiariser avec l'AF tandis que d'autres sont plus abouties. Dans tous les cas, nous avons utilisés des détecteurs disponibles sans chercher à les améliorer ou à les optimiser pour l'application visée.

Ces travaux ont fait l'objet de plusieurs publications nationales et internationales (voir page 159). De plus, ils ont également été utilisés en interne de manière plus approfondie pour la localisation avec notamment les travaux de Claude Aynaud [Aynaud et al., 2014, Aynaud, 2015]. Ces derniers sont actuellement poursuivis par Laurent Delobel [Delobel et al., 2015] où, avec le même principe, une estimation de positionnement exploitant jusqu'à cinq télémètres laser est réalisée.

V.2 Perspectives

Bien que les travaux présentés soient encourageants, ils peuvent être améliorés. L'objectif de cette partie est donc de proposer des pistes qui nous semble pertinentes.

V.2.1 Développement d'une application complète dans le domaine de la reconnaissance d'objets

Bien que plusieurs applications de reconnaissance d'objets aient été présentées, elles n'en sont encore pour la plupart qu'au stade de prototype. De par la généricité de l'AF, il nous semble intéressant de prendre le temps de développer une application complète en apportant notamment un soin particulier au choix des parties et des détecteurs.

En particulier, l'application sur l'estimation fine de pose de véhicule nous parait en bonne voie et demanderait à être poursuivie. Les résultats sur le modèle testé de véhicules sont prometteurs. Il faudrait étendre l'application à d'autres modèles de véhicules. Nous pourrions, peut être, également avoir des détecteurs plus performants

V.2.2 Reconnaissance de scènes

Des applications de reconnaissance d'objets ont été présentées mais il nous semble que l'AF serait bien adapté pour faire de la reconnaissance de scènes. La gestion de plusieurs objets devrait permettre de mettre en valeur la capacité de gestion des occultations de l'AF ainsi que sa capacité de focalisation. De plus, si ces objets partagent des primitives communes, les détections pourraient être mutualisées : un détecteur a renvoyé une roue mais finalement elle n'appartenait pas à l'objet n°1 cherché, elle est mise de côté et pourra servir pour l'objet n°2.

Avant de parvenir à la reconnaissance de scènes, il reste de nombreux problèmes à régler. Il faudrait entre autre être capable de décider s'il vaut mieux estimer précisément un objet avant d'enchaîner sur le suivant ou s'il vaut mieux estimer grossièrement tous les objets avant de les affiner. La première option permet d'être sûr de l'objet trouvé mais en

contrepartie se prive de l'explication de la non détection de certaines de ces parties par l'occultation liée à d'autres objets. La deuxième option a les effets inverses : les occultations relatives entre les objets sont mieux estimées avant d'affiner la position. Par contre, certains objets peuvent malheureusement être erronés du fait d'une mauvaise association non encore corrigée (l'estimation n'étant pas finale).

V.2.3 Changement d'estimateur

Nous avons vu dans les applications de reconnaissance de routes (voir le § IV.4) et d'estimation fine de pose (voir le § IV.5) que lorsqu'il y a des non linéarités dans les fonctions $\mathbf{h}_i$, cela pouvait engendrer quelques difficultés et parfois nécessiter de modifier le vecteur de caractéristiques choisi. Ces problèmes proviennent essentiellement de l'utilisation d'un filtre de Kalman étendu pour calculer la nouvelle estimation compte tenu de l'observation. Il pourrait donc être pertinent de chercher un nouvel estimateur qui serait plus adapté en cas de non linéarité.

V.2.4 Adaptation automatique des α et des β

Nos détecteurs sont caractérisés par α, la probabilité de ne pas détecter alors que la primitive est bien présente, et β, la probabilité de détecter alors qu'il n'y a pas réellement de primitive (voir le § III.4.3). Pour le moment, les valeurs de ces paramètres sont données une fois pour toute lors de la phase d'initialisation. Cependant, il serait intéressant de les faire varier au cours du temps.

Par exemple, ils pourraient être initialisés avec des valeurs grossières puis en fonction des résultats fournis par le réseau bayésien, leurs valeurs seraient affinées au fur et à mesure des détections et ce de manière automatique.

V.2.5 Dépendance entre les détecteurs

Toujours dans le domaine des détecteurs, une piste d'amélioration pourrait être de prendre en compte les dépendances entre les détecteurs. Actuellement, pour une même partie si elle n'est pas détectée avec un détecteur et qu'elle n'est pas détectée non plus avec un autre détecteur, il n'y aura aucun lien entre les deux non détection. Donc soit la probabilité d'existence de la partie, soit la confiance vont grandement diminuer. Néanmoins, il est possible que les non détections soient liées. Pour avoir une confiance plus juste, il serait judicieux de prendre ce phénomène en compte.

Par exemple, les détecteurs de face actuels ont souvent des difficultés similaires. Certains sont plus robustes que d'autres mais si l'éclairage du visage est trop sombre ou s'il y a trop d'occultations, ils auront tous tendance à ne pas répondre. Donc lorsqu'un de ces détecteurs échoue, un autre a de fortes chances d'échouer aussi. Il faudrait donc que la diminution de la confiance soit moindre lors d'une succession de non détection avec des détecteurs similaires. Un événement *les détections sont corrélées* pourrait par exemple être rajouté dans le réseau bayésien.

V.2.6 Ajout d'une étape bottom-up

Actuellement, notre algorithme est top-down. Mais il nous parait intéressant de pouvoir ajouter une étape purement bottom-up afin de gérer des imprévus. Ainsi pour une applica-

tion qui impliquerait le guidage d'un véhicule, même s'il n'y a aucun obstacle théorique sur la route, il est judicieux d'être capable de s'arrêter ou de contourner un obstacle non programmé (comme un chien qui traverse la route subitement).

Un processus bottom-up pourrait également être utile pour accélérer les temps de calculs, par exemple en détectant des informations au plus proche du capteur. Ainsi le processus top-down se concentrerait uniquement sur le problème d'association.

V.2.7 Objectifs dynamiques

Actuellement, les objectifs sont fixés dès le départ. Il pourrait cependant être utile de les faire varier au cours de l'application et potentiellement en fonction des résultats de l'algorithme. Prenons l'exemple d'une application qui doit permettre la localisation d'un véhicule pour envisager un guidage automatique. Si le véhicule est sur une autoroute alors qu'il n'y a pas d'obstacle, il n'est pas très utile d'avoir une précision longitudinale très élevée. Par contre il est impératif de bien rester sur sa file et donc d'avoir une précision latérale élevée. Toutefois, en ville, notamment pour se garer ou s'arrêter à un feu, il faut avoir une bonne précision longitudinale.

Semblablement pour une application de suivi, il pourrait être judicieux d'adapter le niveau de détails souhaité en fonction des conditions. Ainsi si le mouvement est rapide, suivre l'objet sera difficile et nécessitera beaucoup de ressources. Il n'est alors pas indispensable d'exiger une grande précision de reconnaissance. Tandis que si le mouvement est lent, l'objet sera vite retrouvé et donc il sera possible d'affiner la perception de l'objet : déterminer sa couleur, ses contours précis etc.

Publications

[1] Coralie Bernay-Angeletti, Claude Aynaud, Romuald Aufrère, and Roland Chapuis. Stratégie de perception active pour l'interprétation de scènes : Application à une scène routière. In *Orasis, Congrès des jeunes chercheurs en vision par ordinateur*, 2013.

[2] C Bernay-Angeletti, C Aynaud, R Chapuis, and R Aufrère. A top-down perception approach for object recognition. *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, 2014.

[3] Coralie Bernay-Angeletti, Florian Chabot, C Aynaud, R Aufrere, and R Chapuis. A top-down perception approach for vehicle pose estimation. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2240–2245. IEEE, 2015.

[4] C Bernay-Angeletti, R Aufrère, and R Chapuis. Comparaison entre le ransac et un algorithme focalisant pour la reconnaissance d'objets. *RJCIA*, 2014.

[5] Claude Aynaud, Coralie Bernay-Angeletti, Roland Chapuis, Romuald Auirere, and Christophe Debain. Vehicle localization by using a multi-modality top down approach. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 1415–1420. IEEE, 2014. 2

[6] Claude Aynaud, Coralie Bernay-Angeletti, Roland Chapuis, Romuald Aufrere, Christophe Debain, and Nadir Karam. Real-time vehicle localization by using a top-down process. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–6. IEEE, 2014. 2

Bibliographie

[Abdel-Aziz and Karara, 2015] Abdel-Aziz, Y. and Karara, H. (2015). Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry. *Photogrammetric Engineering & Remote Sensing*, 81(2) :103–107. 138

[Achanta et al., 2008] Achanta, R., Estrada, F., Wils, P., and Süsstrunk, S. (2008). Salient region detection and segmentation. In *Computer Vision Systems*, pages 66–75. Springer. 21, 23

[Achanta et al., 2009] Achanta, R., Hemami, S., Estrada, F., and Susstrunk, S. (2009). Frequency-tuned salient region detection. In *Computer vision and pattern recognition, 2009. cvpr 2009. ieee conference on*, pages 1597–1604. IEEE. 20, 21, 23, 37, 42

[Achanta and Süsstrunk, 2010] Achanta, R. and Süsstrunk, S. (2010). Saliency detection using maximum symmetric surround. In *2010 IEEE International Conference on Image Processing*, pages 2653–2656. 23

[Aufrere, 2001] Aufrere, R. (2001). *Reconnaissance et suivi de route par vision artificielle, application à l'aide à la conduite*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II. 132

[Aufrere et al., 2001] Aufrere, R., Chapuis, R., and Chausse, F. (2001). A model-driven approach for real-time road recognition. *Machine Vision and Applications*, 13(2) :95–107. 132

[Aufrère et al., 2000] Aufrère, R., Marmoiton, F., Chapuis, R., Collange, F., and Dérutin, J. P. (2000). Détection de route et suivi de véhicules par vision pour l'acc. *Traitement du signal*, 17(3) :233–248. 45

[Aynaud, 2015] Aynaud, C. (2015). *Localisation précise et fiable de véhicule par approche multi sensorielle*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II. 40, 88, 145, 151, 155

[Aynaud et al., 2014] Aynaud, C., Bernay-Angeletti, C., Chapuis, R., Aufrere, R., Debain, C., and Karam, N. (2014). Real-time vehicle localization by using a top-down process. In *Information Fusion (FUSION), 2014 17th International Conference on*, pages 1–6. IEEE. 45, 66, 155

[Bajcsy, 1988] Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE*, 76(8) :966–1005. 10

[Ballard and Brown, 1992] Ballard, D. H. and Brown, C. M. (1992). Principles of animate vision. *CVGIP : Image Understanding*, 56(1) :3 – 21. Purposive, Qualitative, Active Vision. 10

[Benaskeur, 2002] Benaskeur, A. R. (2002). Consistent fusion of correlated data sources. In *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, volume 4, pages 2652–2656. 72

[Blaschko and Lampert, 2008] Blaschko, M. B. and Lampert, C. H. (2008). Learning to localize objects with structured output regression. In *ECCV*, pages 2–15. Springer Berlin Heidelberg. 26

[Bouchard and Triggs, 2005] Bouchard, G. and Triggs, B. (2005). Hierarchical part-based visual object categorization. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 710–715. 32, 37, 42

[Bourdev and Brandt, 2005] Bourdev, L. and Brandt, J. (2005). Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243. 28, 37, 42, 47

[Bourdev and Malik, 2009] Bourdev, L. and Malik, J. (2009). Poselets : Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. 31, 44

[Bowmaker and Dartnall, 1980] Bowmaker, J. K. and Dartnall, H. (1980). Visual pigments of rods and cones in a human retina. *The Journal of physiology*, 298(1) :501–511. 11

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. http://opencv.org/opencv-3-1.html. 89, 93

[Buschman and Miller, 2007] Buschman, T. J. and Miller, E. K. (2007). Top-Down Versus Bottom-Up Control of Attention in the Prefrontal and Posterior Parietal Cortices. *Science*, 315(5820) :1860–1862. 17

[Caltech Database,] Caltech Database. http://www.vision.caltech.edu/html-files/archive.html. 26

[Carneiro and Lowe, 2006] Carneiro, G. and Lowe, D. (2006). Sparse flexible models of local features. In *Computer Vision–ECCV 2006*, pages 29–43. Springer. xi, 32, 33

[Cerf et al., 2009a] Cerf, M., Frady, E. P., and Koch, C. (2009a). Faces and text attract gaze independent of the task : Experimental data and computer model. *Journal of Vision*, 9(12) :10–10. 35

[Cerf et al., 2008] Cerf, M., Harel, J., Einhäuser, W., and Koch, C. (2008). Predicting human gaze using low-level saliency combined with face detection. In *Advances in neural information processing systems*, pages 241–248. 35, 37

[Cerf et al., 2009b] Cerf, M., Harel, J., Huth, A., Einhäuser, W., and Koch, C. (2009b). Decoding what people see from where they look : Predicting visual stimuli from scanpaths. In *Attention in Cognitive Systems*, pages 15–26. Springer. 35

[Chabot et al., 2015] Chabot, F., Chaouch, M., Rabarisoa, J., Chateau, T., and Teulière, C. (2015). Détection de pose de véhicule pour la reconnaissance de marque et modele. In *Journées francophones des jeunes chercheurs en vision par ordinateur*. 139

[Chapuis et al., 2008] Chapuis, R., Chausse, F., and Trujillo, N. (2008). Progressive Focusing : A Top Down Attentional Vision System. In *Advances in Visual Computing*, volume 5358, pages 468–477. Springer Berlin Heidelberg, Berlin, Heidelberg. 45

[Chen et al., 2006] Chen, H., Xu, Z. J., Liu, Z. Q., and Zhu, S. C. (2006). Composite templates for cloth modeling and sketching. volume 1, pages 943–950. 31, 44

[Chen and Chen, 2008] Chen, Y.-T. and Chen, C.-S. (2008). Fast Human Detection Using a Novel Boosted Cascading Structure With Meta Stages. *IEEE Transactions on Image Processing*, 17(8) :1452–1464. 29

[Cheng et al., 2015] Cheng, M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S. (2015). Global contrast based salient region detection. pages 569–582. 23

[Chenlei Guo and Liming Zhang, 2010] Chenlei Guo and Liming Zhang (2010). A Novel Multiresolution Spatiotemporal Saliency Detection Model and Its Applications in Image and Video Compression. *IEEE Transactions on Image Processing*, 19(1) :185–198. 23, 37

[Cho, 1997] Cho, S.-B. (1997). Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *Neural Networks, IEEE Transactions on*, 8(1) :43–53. 28

[Chum and Matas, 2005] Chum, O. and Matas, J. (2005). Matching with PROSAC-progressive sample consensus. In *IEEE Computer Society Conference on, Computer Vision and Pattern Recognition*, volume 1, page 220–226, San Diego, CA, USA. 24, 113

[Chum et al., 2003] Chum, O., Matas, J., and Kittler, J. (2003). Locally optimized RANSAC. In *Pattern Recognition*, page 236–243. Springer Berlin Heidelberg. 113

[Chum et al., 2005] Chum, O., Werner, T., and Matas, J. (2005). Two-view geometry estimation unaffected by a dominant plane. In *IEEE Computer Society Conference on, Computer Vision and Pattern Recognition*, volume 1, page 772–779, San Diego, CA, USA. 113

[Chun and Wolfe, 2001] Chun, M. and Wolfe, J. (2001). Visual attention. In *Blackwell Handbook of Perception*, pages 2–335. Blackwell. 14

[Clark and Ferrier, 1988] Clark, J. and Ferrier, N. J. (1988). Modal control of an attentive vision system. In *Computer Vision., Second International Conference on*, pages 514–523. 22, 23

[Cootes et al., 1995] Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1995). Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1) :38 – 59. 32

[Craik, 1967] Craik, K. (1967). *The Nature of Explanation*. Philosophy : science CAM. Cambridge University Press. 10

[Crandall et al., 2005] Crandall, D., Felzenszwalb, P., and Hutten, D. (2005). Spatial priors for part-based recognition using statistical models. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 10–17. IEEE. 32

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines : And Other Kernel-based Learning Methods*. Cambridge University Press, New York, USA. 26, 37

[Csurka et al., 2004] Csurka, G., Dance, C., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague. 26

[Dahab et al., 2012] Dahab, D. A., Ghoniemy, S. S., and Selim, G. M. (2012). Automated Brain Tumor Detection and Identification Using Image Processing and Probabilistic Neural Network Techniques. *International journal of image processing and visual communication*, 1(2) :1–8. 28

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE. 27

[Dayan et al., 2000] Dayan, P., Kakade, S., and Montague, P. R. (2000). Learning and selective attention. *nature neuroscience*, 3 :1218–1223. 18

[DeCarlo and Santella, 2002] DeCarlo, D. and Santella, A. (2002). Stylization and abstraction of photographs. In *ACM transactions on graphics (TOG)*, volume 21, pages 769–776. ACM. 23

[Delobel et al., 2015] Delobel, L., Aynaud, C., Aufrere, R., Debain, C., Chapuis, R., Chateau, T., and Bernay-Angeletti, C. (2015). Robust localization using a top-down approach with several lidar sensors. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2371–2376. 155

[Delorme and Flückiger, 2003] Delorme, A. and Flückiger, M. (2003). *Perception et réalité : Introduction à la psychologie des perceptions*. Neurosciences & cognition. De Boeck Supérieur. 13, 14

[Descamps et al., 2011] Descamps, S., Béchet, A., Descombes, X., Arnaud, A., and Zerubia, J. (2011). An automatic counter for aerial images of aggregations of large birds. *Bird study*, 58(3) :302–308. 8

[Dollár et al., 2012] Dollár, P., Appel, R., and Kienzle, W. (2012). Crosstalk cascades for frame-rate pedestrian detection. In *Computer Vision–ECCV 2012*, pages 645–659. Springer. 28, 37, 42, 47

[Draper and Lionelle, 2003] Draper, B. and Lionelle, A. (2003). Evaluation of selective attention under similarity transforms. In *International Workshop on Attention and Performance in Computer Vision*, pages 31–38. 22

[Ehinger et al., 2009] Ehinger, K. A., Hidalgo-Sotelo, B., Torralba, A., and Oliva, A. (2009). Modelling search for people in 900 scenes : A combined source model of eye guidance. *Visual Cognition*, 17(6-7) :945–978. 34

[Einhauser et al., 2008] Einhauser, W., Spain, M., and Perona, P. (2008). Objects predict fixations better than early saliency. *Journal of Vision*, 8. 35

[Elazary and Itti, 2010] Elazary, L. and Itti, L. (2010). A Bayesian model for efficient visual search and recognition. *Vision Research*, 50(14) :1338–1352. 35, 38

[Fei-Fei et al., 2003] Fei-Fei, L., Fergus, R., and Perona, P. (2003). A Bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. 32, 37, 42

[Felzenszwalb et al., 2008] Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. 31, 33

[Felzenszwalb et al., 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9) :1627–1645. 40, 137

[Felzenszwalb and Huttenlocher, 2005] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1) :55–79. 32

[Felzenszwalb and McAllester, 2011] Felzenszwalb, P. F. and McAllester, D. (2011). Object detection grammars. page 691. 31, 37, 42, 44, 47

[Fergus et al., 2003] Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. IEEE Computer Society Conference on*, volume 2, pages II–264. 32, 37, 42, 47

[Fergus et al., 2006] Fergus, R., Perona, P., and Zisserman, A. (2006). A sparse object category model for efficient learning and complete recognition. *Toward category-level object recognition*, pages 443–461. 31

[Fernandez, 2010] Fernandez, D. (2010). *L'attention visuelle sélective : pertinence, saillance, résistance à l'interférence*. PhD thesis, Lyon 2. 13

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395. 24, 112

[Fitts, 1992] Fitts, P. M. (1992). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology : General*, 121(3) :262. 12

[Gall and Lempitsky, 2009] Gall, J. and Lempitsky, V. (2009). Class-specific hough forests for object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1022–1029. 31, 33

[Gao et al., 2008] Gao, D., Mahadevan, V., and Vasconcelos, N. (2008). On the plausibility of the discriminant center-surround hypothesis for visual saliency. *Journal of vision*, 8(7) :13. 35, 44

[Gao and Vasconcelos, 2004] Gao, D. and Vasconcelos, N. (2004). Discriminant saliency for visual recognition from cluttered scenes. In *Advances in neural information processing systems*, pages 481–488. 35

[Gao and Vasconcelos, 2005] Gao, D. and Vasconcelos, N. (2005). Integrated learning of saliency, complex features, and object detectors from cluttered scenes. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 282–287 vol. 2. 35, 38

[Gibson, 1962] Gibson, J. J. (1962). Observations on active touch. *Psychological review*, 69(6) :477. 10

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE. 137

[Goferman et al., 2012] Goferman, S., Zelnik-Manor, L., and Tal, A. (2012). Context-aware saliency detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10) :1915–1926. 34

[Goldberg and Kotval, 1999] Goldberg, J. H. and Kotval, X. P. (1999). Computer interface evaluation using eye movements : methods and constructs. *International Journal of Industrial Ergonomics*, 24(6) :631–645. 12

[Goutte et al., 2004] Goutte, C., Gaussier, E., Cancedda, N., and Déjean, H. (2004). Generative vs discriminative approaches to entity recognition from label-deficient data. *JADT Journée internationales d'Analyse statistique des Données Textuelles*, pages 515–523. 26

[Grabli et al., 2004] Grabli, S., Durand, F., and Sillion, F. X. (2004). Density measure for line-drawing simplification. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, pages 309–318. 23, 37

[Gregory et al., 2000] Gregory, R., Mattheeuws-Hambrouck, M., and Thinès, G. (2000). *L'œil et le cerveau : La psychologie de la vision*. Neurosciences & cognition. De Boeck Supérieur. 11

[Harel et al., 2007] Harel, J., Koch, C., and Perona, P. (2007). Graph-based visual saliency. pages 545–552. MIT Press. 21, 22

[Henderson et al., 2007] Henderson, J. M., Brockmole, J. R., Castelhano, M. S., and Mack, M. (2007). Visual saliency does not account for eye movements during visual search in real-world scenes. *Eye movements : A window on mind and brain*, pages 537–562. 34

[Henderson and Feiner, 2009] Henderson, S. J. and Feiner, S. (2009). Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 135–144. 9

[Hong et al., 2015] Hong, S., You, T., Kwak, S., and Han, B. (2015). Online tracking by learning discriminative saliency map with convolutional neural network. *arXiv preprint arXiv :1502.06796*. 23

[Hou and Zhang, 2007] Hou, X. and Zhang, L. (2007). Saliency detection : A spectral residual approach. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 21, 23

[Hsiao et al., 2014] Hsiao, E., Sinha, S. N., Ramnath, K., Baker, S., Zitnick, L., and Szeliski, R. (2014). Car make and model recognition using 3d curve alignment. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*. IEEE. 137

[Itti, 2004] Itti, L. (2004). Automatic Foveation for Video Compression Using a Neurobiological Model of Visual Attention. *IEEE Transactions on Image Processing*, 13(10) :1304–1318. 23

[Itti and Koch, 2000] Itti, L. and Koch, C. (2000). A Saliency-Based Search Mechanism for Overt and Covert Shifts of Visual Attention. *Vision research*, 40 :1489–1506. 22

[Itti and Koch, 2001] Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature reviews neuroscience*, 2 :194–203. xi, 20, 21, 22

[Itti et al., 1998] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11) :1254–1259. 20, 21, 22, 35, 37, 42

[Jain et al., 2000] Jain, A. K., Duin, R. P., and Mao, J. (2000). Statistical pattern recognition : A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1) :4–37. 25, 37

[Jindal and Psounis, 2004] Jindal, A. and Psounis, K. (2004). Modeling spatially-correlated sensor network data. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 162–171. 72

[Kanan et al., 2009] Kanan, C., Tong, M. H., Zhang, L., and Cottrell, G. W. (2009). SUN : Top-down saliency using natural statistics. *Visual Cognition*, 17(6-7) :979–1003. 34

[King, 2009] King, D. E. (2009). Dlib-ml : A machine learning toolkit. *Journal of Machine Learning Research*, 10 :1755–1758. 89, 120, 151, XIII

[Kirby and Sirovich, 1990] Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loeve procedure for the characterization of human faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1) :103–108. 27

[Koch and Ullman, 1985] Koch, C. and Ullman, S. (1985). Shifts in selective visual attention : towards the underlying neural circuitry. *Human Neurobiology*, 4 :219–227. 20, 22, 23

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 28, 37

[Lazebnik et al., 2006] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. 31

[Le Meur et al., 2005] Le Meur, O., Barba, D., Le Callet, P., de Nantes, U., de Nantes. Faculté des sciences et des techniques, U., École doctorale sciences et technologies de l'information et des matériaux (Nantes), École centrale de Nantes, and École nationale supérieure des mines (Nantes) (2005). *Attention sélective en visualisation d'images fixes et animées affichées sur écran*. PhD thesis, [s.n.], [S.l.]. 23, 37, 42

[Le Meur et al., 2006] Le Meur, O., Le Callet, P., Barba, D., and Thoreau, D. (2006). A coherent computational approach to model bottom-up visual attention. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5) :802–817. 20, 23

[Lecas, 1992] Lecas, J. (1992). *L'attention visuelle : de la conscience aux neurosciences*. Psychologie et sciences humaines. Mardaga. 12

[Legrand, 1972] Legrand, Y. (1972). *Optique physiologique. Tome 2 : lumière et couleurs*, volume 2. Masson Paris, 2ième édition edition. 11

[Leibe et al., 2004] Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7. 31, 33, 37

[Leibe et al., 2008] Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3) :259–289. 31, 33

[Lepetit et al., 2009] Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2) :155–166. 138

[Leung et al., 1995] Leung, T. K., Burl, M. C., and Perona, P. (1995). Finding faces in cluttered scenes using random labeled graph matching. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 637–644. IEEE. 30, 37, 42, 44, 47

[LFW Face Database,] LFW Face Database. http://vis-www.cs.umass.edu/lfw/. 26

[Lim et al., 2014] Lim, J. J., Khosla, A., and Torralba, A. (2014). Fpm : Fine pose parts-based model with 3d cad models. In *Computer Vision–ECCV 2014*, pages 478–493. Springer. 137

[Lin et al., 2009] Lin, L., Wu, T., Porway, J., and Xu, Z. (2009). A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition*, 42 :1297 – 1307. 31

[Liu et al., 2007] Liu, T., Sun, J., Zheng, N.-N., Tang, X., and Shum, H.-Y. (2007). Learning to detect a salient object. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8. 23

[Lowe, 1991] Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*. 137

[Ma and Zhang, 2003] Ma, Y.-F. and Zhang, H.-J. (2003). Contrast-based image attention analysis by using fuzzy growing. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA '03, pages 374–381, New York, USA. ACM. 21

[Marchesotti et al., 2009] Marchesotti, L., Cifarelli, C., and Csurka, G. (2009). A framework for visual saliency detection with applications to image thumbnailing. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2232–2239. 23, 35

[McLachlan, 2004] McLachlan, G. J. (2004). *Discriminant analysis and statistical pattern recognition*. Wiley series in probability and statistics. Hoboken, N.J. 26

[Milanese et al., 1994] Milanese, R., Wechsler, H., Gill, S., Bost, J.-M., and Pun, T. (1994). Integration of bottom-up and top-down cues for visual attention using non-linear relaxation. In *Computer vision and pattern recognition, 1994 ieee computer society conference on*, pages 781–785. 22

[Most et al., 2005] Most, S. B., Scholl, B. J., Clifford, E. R., and Simons, D. J. (2005). What You See Is What You Set : Sustained Inattentional Blindness and the Capture of Awareness. *Psychological Review*, 112(1) :217–242. 13

[Mundy, 2006] Mundy, J. L. (2006). Object recognition in the geometric era : A retrospective. In *Toward category-level object recognition*, pages 3–28. Springer. 137

[Navalpakkam and Itti, 2006] Navalpakkam, V. and Itti, L. (2006). An integrated model of top-down and bottom-up attention for optimizing detection speed. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2049–2056. 22, 37, 42

[Navalpakkam and Itti, 2007] Navalpakkam, V. and Itti, L. (2007). Search Goal Tunes Visual Features Optimally. *Neuron*, 53(4) :605–617. 35

[Ng and Jordan, 2002] Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press. 26

[Nothdurft, 2000] Nothdurft, H.-C. (2000). Salience from feature contrast : Variations with texture density. *Vision Research*, 40(23) :3181–3200. 13

[Oliva et al., 2003] Oliva, A., Torralba, A., Castelhano, M. S., and M.Henderson, J. (2003). Top-down control of visual attention in object detection. In *IEEE Proceedings of the International Conference on Image Processing*, volume 1, pages 253–256. 34

[Oquab et al., 2014] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1717–1724. 28

[Papageorgiou et al., 1998] Papageorgiou, C. P., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. 29, 38

[Parhizkar et al., 2011] Parhizkar, B., Al-Modwahi, A. A. M., Lashkari, A. H., Bartaripou, M. M., and Babae, H. R. (2011). A Survey on Web-based AR Applications. *arXiv preprint arXiv :1111.2993*. 10

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 102, 103

[Pezzulo, 2008] Pezzulo, G. (2008). Coordinating with the future : The anticipatory nature of representation. *Minds and Machines*, 18(2) :179–225. 10

[Posner, 1980] Posner, M. I. (1980). Orienting of attention. *Quarterly Journal of Experimental Psychology*, 32(1) :3–25. 13

[Prasad, 2012] Prasad, D. K. (2012). Survey of the problem of object detection in real images. *International Journal of Image Processing (IJIP)*. 7

[Prescott et al., 2011] Prescott, T. J., Diamond, M. E., and Wing, A. M. (2011). Active touch sensing. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 366(1581) :2989–2995. 10

[Privitera and Stark, 2000] Privitera, C. M. and Stark, L. W. (2000). Algorithms for defining visual regions-of-interest : Comparison with eye fixations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(9) :970–982. 23

[Prokaj and Medioni, 2009] Prokaj, J. and Medioni, G. (2009). 3-D model based vehicle recognition. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–7. IEEE. 137

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1) :81–106. 26

[Quinton et al., 2014] Quinton, J.-C., Catenacci, N., Barca, L., and Pezzulo, G. (2014). The cat is on the mat. or is it a dog ? dynamic competition in perceptual decision making. *IEEE Transactions on Systems, Man and Cybernetics : Systems*, 44 :539–551. 10

[Rabin et al., 2010] Rabin, J., Delon, J., Gousseau, Y., and Moisan, L. (2010). Mac-Ransac : reconnaissance automatique d'objets multiples. *Actes du congrès Reconnaissance de Formes et Intelligence Artificielle (RFIA), Caen, France*. 24, 113

[Ramírez et al., 2013] Ramírez, J., Górriz, J., Salas-Gonzalez, D., Romero, A., López, M., Álvarez, I., and Gómez-Río, M. (2013). Computer-aided diagnosis of Alzheimer's type dementia combining support vector machines and discriminant set of features. *Information Sciences*, 237 :59–72. 9

[Riesenhuber and Poggio, 1999] Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11) :1019–1025. 14

[Rodieck, 2003] Rodieck, R. (2003). *La vision*. Neurosciences & cognition. De Boeck Supérieur. 12

[Sahli et al., 2012] Sahli, S., Lavigne, D. A., and Sheng, Y. (2012). Saliency detection in aerial imagery using multi-scale SLIC segmentation. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). 22, 23

[Sankari and Adeli, 2011] Sankari, Z. and Adeli, H. (2011). Probabilistic neural networks for diagnosis of alzheimer's disease using conventional and wavelet coherence. *Journal of Neuroscience Methods*, 197(1) :165 – 170. 28, 37, 42, 47

[Sattler et al., 2009] Sattler, T., Leibe, B., and Kobbelt, L. (2009). SCRAMSAC : improving RANSAC's efficiency with a spatial consistency filter. In *IEEE 12th International Conference on, Computer Vision*, page 2090–2097, Kyoto, Japan. 24, 113

[Schölkopf and Smola, 2002a] Schölkopf, B. and Smola, A. J. (2002a). *Learning with kernels : Support vector machines, regularization, optimization, and beyond*. MIT press. 25, 37, 42

[Schölkopf and Smola, 2002b] Schölkopf, B. and Smola, A. J. (2002b). *Learning with kernels : Support vector machines, regularization, optimization, and beyond*. MIT press. 26

[Si and Zhu, 2012] Si, Z. and Zhu, S. C. (2012). Learning and-or templates for object recognition and detection. Technical report, Technical report, Statistics Dept., UCLA, 2011. 4. 31, 37, 44

[Siagian and Itti, 2009] Siagian, C. and Itti, L. (2009). Biologically inspired mobile robot vision localization. *Robotics, IEEE Transactions on*, 25(4) :861–873. 23, 37

[Simon Julier and JeffreyK Uhlmann, 2001] Simon Julier and JeffreyK Uhlmann (2001). General Decentralized Data Fusion with Covariance Intersection (CI). In *Multisensor Data Fusion*, Electrical Engineering & Applied Signal Processing Series. CRC Press. 72

[Simons and Chabris, 1999] Simons, D. J. and Chabris, C. F. (1999). Gorillas in our midst : Sustained inattentional blindness for dynamic events. *Perception-London*, 28(9) :1059–1074. 13

[Simoës-Perlant and Largy, 2008] Simoës-Perlant, A. and Largy, P. (2008). L'apprentissage implicite chez l'enfant présentant des troubles du langage écrit. *ANAE*, 96 :27–32. 24

[Sobottka and Pitas, 1996] Sobottka, K. and Pitas, I. (1996). Looking for faces and facial features in color images. In *PRIA : Advances in Mathematical Theory and Applications*. 29, 38

[Tarr, 2000] Tarr, M. J. (2000). Visual pattern recognition. *Encyclopedia of psychology*, pages 1–4. 10

[Tessier et al., 2009] Tessier, C., Debain, C., Chapuis, R., and Chausse, F. (2009). Map aided localization and vehicle guidance using an active landmark search. *International Journal on Information Fusion*. 45, 54

[The PASCAL Visual Object Classes,] The PASCAL Visual Object Classes. `http://host.robots.ox.ac.uk/pascal/VOC/`. 26

[Torralba et al., 2006] Torralba, A., Oliva, A., Castelhano, M. S., and Henderson, J. M. (2006). Contextual guidance of eye movements and attention in real-world scenes : The role of global features in object search. *Psychological Review*, 113(4) :766–786. 34

[Treisman and Gelade, 1980] Treisman, A. M. and Gelade, G. (1980). A Feature-Integration theory of attention. *Cognitive Psychology*, 12(1) :97–136. 20, 37

[Trujillo Morales, 2007] Trujillo Morales, N. (2007). *Stratégie de perception pour la compréhension de scènes par une approche focalisante, application à la reconnaissance d'objets*. PhD thesis, Clermont-Ferrand 2. 45

[Tsochantaridis et al., 2004] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*. ACM. 26

[Turk and Pentland, 1991] Turk, M. and Pentland, A. (1991). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591. 27

[Urban et al., 2011] Urban, F., Follet, B., Chamaret, C., Le, O., Baccino, M. T., Urban, F., Chamaret, B. F. C., Follet, B., Chamaret, C., Meur, O. L., Follet, B., and Baccino, T. (2011). *Medium Spatial Frequencies, a Strong Predictor of Salience*. 42

[Urban et al., 2010] Urban, F., Follet, B., Chamaret, C., Meur, O., and Baccino, T. (2010). Medium spatial frequencies, a strong predictor of salience. *Cognitive Computation*, 3(1) :37–47. 23, 37

[Valenti et al., 2009] Valenti, R., Sebe, N., and Gevers, T. (2009). Isocentric color saliency in images. pages 993–996. 23, 37

[Valera and Velastin, 2005] Valera, M. and Velastin, S. (2005). Intelligent distributed surveillance systems : a review. *IEE Proceedings - Vision, Image, and Signal Processing*, 152(2) :192. 9

[Van Nguyen et al., 2013] Van Nguyen, H., Patel, V. M., Nasrabadi, N. M., and Chellappa, R. (2013). Design of non-linear kernel dictionaries for object recognition. *Image Processing, IEEE Transactions on*, 22(12) :5123–5135. 27

[Vapnik, 1999] Vapnik, V. N. (1999). An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5) :988–999. 26

[Viana et al., 2013] Viana, I., Parlouar, R., Bugarin, F., Orteu, J.-J., and Brethes, L. (2013). Inspection automatisée d'assemblages mécaniques : vers une approche couplée vision 2d/vision 3d. 8

[Vikram et al., 2011a] Vikram, T. N., Tscherepanow, M., and Wrede, B. (2011a). A random center surround bottom up visual attention model useful for salient region detection. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 166–173. IEEE. 22

[Vikram et al., 2011b] Vikram, T. N., Tscherepanow, M., and Wrede, B. (2011b). A visual saliency map based on random sub-window means. In *Pattern Recognition and Image Analysis*, pages 33–40. Springer. 22

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. 27, 29, 38, 40, 70

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2) :137–154. xi, 27, 28, 37, 42, 47

[Vivet, 2011] Vivet, D. (2011). *Perception de l'environnement par radar hyperfréquence. Application à la localisation et la cartographie simultanées, à la détection et au suivi d'objets mobiles en milieu extérieur*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II. 8

[Wan and Van Der Merwe, 2000] Wan, E. A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee. 53

[Wolfe et al., 1989] Wolfe, J. M., Cave, K. R., and Franzel, S. L. (1989). Guided search : an alternative to the feature integration model for visual search. *Journal of Experimental Psychology : Human perception and performance*, 15(3) :419. 20, 34, 37, 44

[Wolfe and Horowitz, 2004] Wolfe, J. M. and Horowitz, T. S. (2004). What attributes guide the deployment of visual attention and how do they do it ? *Nature Reviews Neuroscience*, 5(6) :495–501. 13

[Xie and Lu, 2011] Xie, Y. and Lu, H. (2011). Visual saliency detection based on Bayesian model. pages 645–648. IEEE. 22

[Yarbus, 1967] Yarbus, A. (1967). Eye movements during perception of complex objects. In *Eye Movements and Vision*, pages 171–211. Springer US. 15, 16, 34

[Yssaad, 2001] Yssaad, R. (2001). *Analyse psychophysique du champ visuel. — Détection, Identification, Effet de groupement et Apprentissage Perceptif*. PhD thesis, Université Lumière Lyon 2. 11

[Zhang et al., 2015] Zhang, D., Li, W., Sun, M., and Yu, H. (2015). Saliency Map for Object Tracking. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(10) :233–240. 23

[Zhang, 2004] Zhang, H. (2004). The optimality of naive Bayes. *AA*, 1(2) :3. 26

[Zhang et al., 2008] Zhang, L., Tong, M. H., Marks, T. K., Shan, H., and Cottrell, G. W. (2008). SUN : A Bayesian framework for saliency using natural statistics. *Journal of vision*, 8(7). 34

[Zhu and Mumford, 2006] Zhu, S.-C. and Mumford, D. (2006). A Stochastic Grammar of Images. volume 2, pages 259–362. 31, 37, 44

[Zia et al., 2011] Zia, M. Z., Stark, M., Schiele, B., and Schindler, K. (2011). Revisiting 3d geometric models for accurate object shape and pose. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 569–576. 137

Annexe A

Matrices de covariance et bornes selon les axes

Nous cherchons, ici, à définir les bornes maximum et minimum selon chaque axe pour un ellipsoïde centré sur l'origine. La figure A.1 est un exemple en deux dimensions. Nous avons une ellipse centrée et nous voulons déterminer les valeurs $b_{x_{min}}$, $b_{x_{max}}$, $b_{y_{min}}$ et $b_{y_{max}}$. Comme, par symétrie, $b_{x_{min}} = -b_{x_{max}}$ et $b_{y_{min}} = -b_{y_{max}}$, il suffit de déterminer les bornes maximum selon chaque axe. Il en ira de même en dimension N.

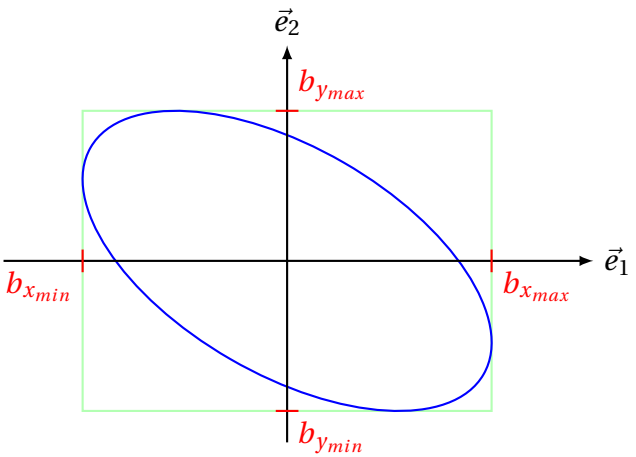


FIGURE A.1 – Exemple des bornes désirées pour une ellipse. Les valeurs $b_{x_{min}}$, $b_{x_{max}}$, $b_{y_{min}}$, $b_{y_{max}}$ permettent de définir la boîte englobante (en vert) de l'ellipse bleue.

Notons A la matrice représentative du contour de l'ellipsoïde. C'est donc une matrice symétrique réelle définie positive. Elle est par conséquent inversible et son inverse sera noté A^{-1}. A^{-1} peut être vue comme la matrice de covariance associée à l'ellipsoïde. Notons $\vec{e}_1, \ldots, \vec{e}_N$ les vecteurs de la base canonique de $\mathbb{R}^N$. Nous cherchons la boîte englobante de l'ellipsoïde A, ce qui revient à chercher pour chaque axe $\vec{e}_i$, un point x appartenant à l'ellipse tel que :

$$x^T A x = 1 \quad : \text{le point appartient à l'ellipse} \tag{A.1}$$

$$\forall j \in [1, N] \text{ tel que } i \neq j \quad \nabla(x^T A x - 1).\vec{e}_j = 0 \quad : x \text{ est un maxima selon } \vec{e}_j \tag{A.2}$$

où ∇ désigne l'opérateur gradient. Donc :

$$\nabla(x^{\mathrm{T}}\mathrm{A}x - 1) = \mathrm{A}x \tag{A.3}$$

Le maximum $b_{i_{max}}$ sur l'axe $\vec{e}_i$ sera alors :

$$b_{i_{max}} = x.\vec{e}_i \tag{A.4}$$

Le minimum $b_{i_{min}}$ pourrait être obtenu similairement en changeant les signes. Nous allons donc chercher dans un premier temps à déterminer la valeur de x et nous pourrons ensuite en déduire les valeurs des $b_{i_{max}}$.

Posons λ_i un réel tel que $\lambda_i \neq 0$. En utilisant la formule (A.3), l'équation (A.2) peut être réécrite :

$$\mathrm{A}x = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \lambda_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i_{eme} \text{ ligne} \tag{A.5}$$

$$\Leftrightarrow \quad \mathrm{A}x = \lambda_i \vec{e}_i \tag{A.6}$$

$$\Leftrightarrow \quad x = \lambda_i \mathrm{A}^{-1} \vec{e}_i \tag{A.7}$$

Pour la suite des calculs, nous utiliserons les notations suivantes :

— $\mathrm{M}_{:,i}$ désignera la i_{eme} colonne de la matrice M.

— $\mathrm{M}_{i,:}$ désignera la i_{eme} ligne de la matrice M.

— $\mathrm{M}_{i,j} = \mathrm{M}(i,j)$ désignera le coefficient de M à la ligne i, colonne j.

Avec ces notations, l'équation (A.7) devient :

$$x = \lambda_i \mathrm{A}^{-1}_{:,i} \tag{A.8}$$

En injectant, la relation (A.8) dans l'équation (A.1) :

$$\lambda_i^2 \, (\mathrm{A}^{-1}_{:,i})^{\mathrm{T}} \, \mathrm{A} \mathrm{A}^{-1}_{:,i} = 1 \tag{A.9}$$

Or

$$\mathrm{A}\mathrm{A}^{-1}_{:,i} = \vec{e}_i \tag{A.10}$$

où I désigne la matrice identité . En effet, $\mathrm{A}\mathrm{A}^{-1}_{:,i}$ donne la i_{eme} colonne de la matrice I qui est égale à $\vec{e}_i$. Donc :

$$\lambda_i^2 \, (\mathrm{A}^{-1}_{:,i})^{\mathrm{T}} \, \vec{e}_i = 1 \tag{A.11}$$

$$\Leftrightarrow \lambda_i^2 \, \mathrm{A}^{-1}_{i,i} = 1 \tag{A.12}$$

$$\tag{A.13}$$

Or

$$\forall i, \; \mathrm{A}^{-1}_{i,i} \neq 0 \tag{A.14}$$

car sinon le déterminant serait nul et donc la matrice ne serait pas inversible. Donc

$$\lambda_i^2 = \frac{1}{A_{i,i}^{-1}} \tag{A.15}$$

$$\Leftrightarrow \quad \lambda_i = \pm\frac{1}{\sqrt{A_{i,i}^{-1}}} \tag{A.16}$$

$$\tag{A.17}$$

En injectant la relation (A.17) dans l'équation (A.8) :

$$x = \frac{\pm 1}{\sqrt{A_{i,i}^{-1}}} A_{:,i}^{-1} \tag{A.18}$$

Nous venons d'exprimer x. Nous pouvons donc reprendre l'équation (A.4) pour obtenir les $b_{i_{max}}$:

$$b_{i_{max}} = x.\vec{e_i} \tag{A.19}$$

$$\Leftrightarrow b_{i_{max}} = \frac{\pm 1}{\sqrt{A_{i,i}^{-1}}} A_{:,i}^{-1}\vec{e_i} \tag{A.20}$$

$$\Leftrightarrow b_{i_{max}} = \frac{\pm A_{i,i}^{-1}}{\sqrt{A_{i,i}^{-1}}} \tag{A.21}$$

$$\Leftrightarrow b_{i_{max}} = \pm\sqrt{A_{i,i}^{-1}} \tag{A.22}$$

$b_{i_{max}}$ représente la borne maximum selon l'axe $\vec{e_i}$ donc

$$b_{i_{max}} = \sqrt{A_{i,i}^{-1}} \tag{A.23}$$

Nous avons donc déterminé selon chaque axe, la valeur de la borne maximum.

Annexe B

Filtre de Kalman

B.1 Introduction

Le filtre de Kalman est un estimateur assez populaire qui à partir d'observations successives (supposées bruitées mais avec un bruit gaussien) cherche à estimer les paramètres d'un système. Usuellement l'ensemble des paramètres du système est appelé **état du système** noté $\underline{X}$. La matrice de covariance associée est notée **P**.

Le filtre de Kalman a deux phases distinctes : Prédiction et Mise à jour. La phase de prédiction utilise l'état estimé de l'instant précédent pour produire une estimation de l'état courant. Dans l'étape de mise à jour, les observations de l'instant courant sont utilisées pour corriger l'état prédit dans le but d'obtenir une estimation plus précise.

Ce filtre est un estimateur récursif : l'état à l'étape $k+1$ dépend uniquement de l'état à l'instant k et et des observations à l'instant $k+1$. Un schéma de fonctionnement du filtre de Kalman est proposé dans la figure B.1. Le filtre de Kalman est simple à mettre en place, donne de bons résultats pour les systèmes linéaires et est peu coûteux en terme de temps de calcul.

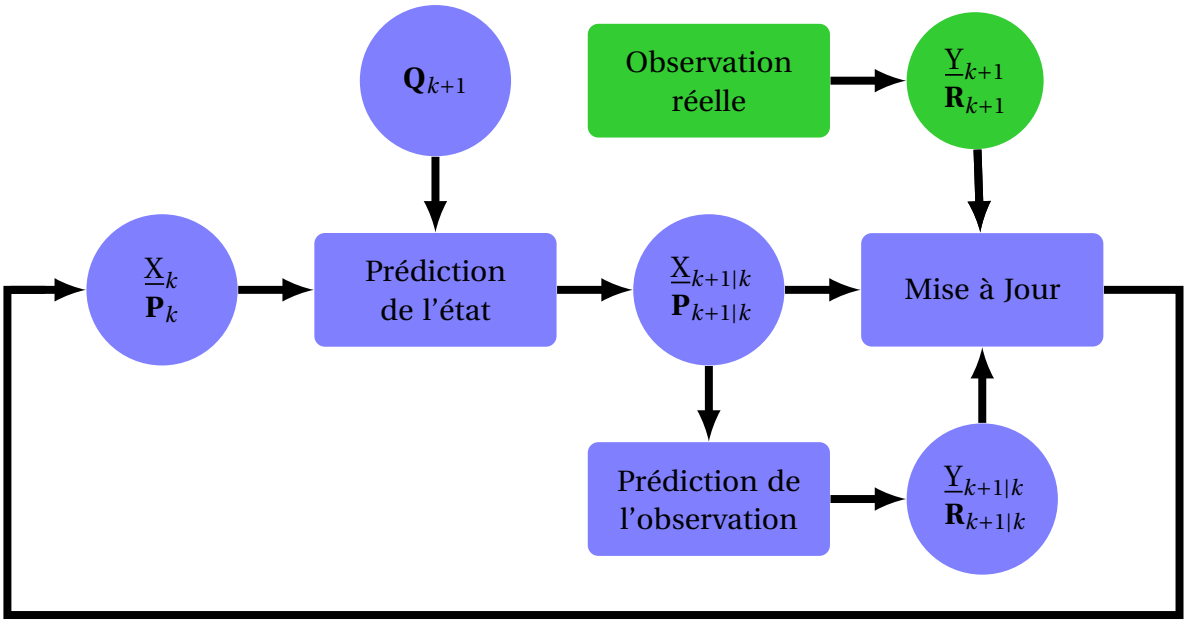


FIGURE B.1 – Schéma de principe du filtre de Kalman.

B.2 Filtre de Kalman Linéaire

B.2.1 Modèle d'observation

Le modèle d'observation ou équation de mesure décrit la dépendance des observations avec le vecteur de paramètres $\underline{X}_k$:

$$\underline{Y}_k = \mathbf{H}_k \underline{X}_k + \underline{V}_k$$

où

— $\underline{Y}_k$: observation ou mesure du processus à l'instant k

— $\mathbf{H}_k$: matrice de mesure.
C'est la matrice qui relie l'état à la mesure

— $\underline{V}_k$: bruit de mesure

B.2.2 Modèle d'évolution

Le modèle d'évolution décrit la dynamique du vecteur d'état $\underline{X}_k$

$$\underline{X}_{k+1} = \mathbf{F}_{k+1} \underline{X}_k + \underline{U}_k$$

où

— $\underline{X}_{k+1}$: état à l'instant $k+1$

— $\mathbf{F}_{k+1}$: matrice d'évolution.
C'est la matrice qui relie l'état actuel k à l'état suivant $k+1$

— $\underline{U}_k$: bruit d'évolution

B.2.3 Prédiction

$$\begin{aligned} \underline{\mathrm{X}}_{k+1|k} &= \mathbf{F}_{k+1}\underline{\mathrm{X}}_k \\ \mathbf{P}_{k+1|k} &= \mathbf{F}_{k+1}\mathbf{P}_k\mathbf{F}_{k+1}^{\mathrm{T}} + \mathbf{Q}_k \end{aligned} \tag{B.1}$$

avec

— $\mathbf{Q}_k$: matrice de covariance du bruit d'évolution

B.2.4 Mise à Jour

$$\begin{aligned} \underline{\mathrm{Y}}_{k+1|k} &= \mathbf{H}_k\underline{\mathrm{X}}_{k+1|k} && \text{(mesure prédite)} \\ \underline{\mathrm{Z}}_{k+1} &= \underline{\mathrm{Y}}_{k+1} - \underline{\mathrm{Y}}_{k+1|k} && \text{(innovation)} \\ \mathbf{S}_{k+1} &= \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^{\mathrm{T}} + \mathbf{R}_{k+1} && \text{(covariance de l'innovation)} \\ \mathrm{K}_{k+1} &= \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{S}_{k+1}^{-1} && \text{(gain de Kalman optimal)} \\ \underline{\mathrm{X}}_{k+1} &= \underline{\mathrm{X}}_{k+1|k} + \mathrm{K}_{k+1}\underline{\mathrm{Z}}_{k+1} && \text{(état mis à jour)} \\ \mathbf{P}_{k+1} &= (\mathrm{I} - \mathrm{K}_{k+1}\mathbf{H}_{k+1})\mathbf{P}_{k+1|k} && \text{(covariance mise à jour)} \end{aligned} \tag{B.2}$$

avec

— I : matrice identité aux dimensions adéquates

— $\mathbf{R}_{k+1}$: matrice de covariance du bruit de mesure

Ce filtre de Kalman est limité aux systèmes linéaires. Cependant, la plupart des systèmes physiques sont non linéaires. La non-linéarité peut être associée au modèle du processus, au modèle d'observation ou bien aux deux. Il existe donc un autre modèle du filtre de Kalman pour pallier ce problème.

B.3 Filtre de Kalman Étendu (EKF)

B.3.1 Modèles d'évolution et d'observation

Dans le filtre de Kalman étendu (EKF), les modèles d'évolution et d'observation n'ont pas besoin d'être des fonctions linéaires de l'état mais peuvent à la place être des fonctions différentiables

$$\begin{aligned} \underline{\mathrm{X}}_{k+1} &= \mathbf{f}(\underline{\mathrm{X}}_k, k+1) + \underline{\mathrm{U}}_k \\ \underline{\mathrm{Y}}_{k+1} &= \mathbf{h}(\underline{\mathrm{X}}_{k+1|k}, k) + \underline{\mathrm{V}}_k \end{aligned} \tag{B.3}$$

La fonction **f** est utilisée pour calculer l'état prédit à partir de l'état estimé précédent. La fonction **h** est employée pour calculer l'observation prédite de l'état prédit. Cependant, **f** et **h** ne peuvent pas être appliquées directement au calcul de la covariance : leur jacobienne est utilisée. À chaque instant k, la jacobienne est évaluée avec les états estimés courants. Ce processus linéarise essentiellement la fonction non linéaire autour de l'estimation courante.

B.3.2 Prédiction

$$\begin{aligned} \underline{\mathrm{X}}_{k+1|k} &= \mathbf{f}(\underline{\mathrm{X}}_k, k+1) \\ \mathbf{P}_{k+1|k} &= \mathbf{F}_{k+1}\mathbf{P}_k\mathbf{F}_{k+1}^{\mathrm{T}} + \mathbf{Q}_k \end{aligned} \tag{B.4}$$

B.3.3 Mise à Jour

$$\begin{aligned} \underline{\mathrm{Y}}_{k+1|k} &= \mathbf{h}(\underline{\mathrm{X}}_{k+1|k}, k+1) \\ \underline{\mathrm{Z}}_{k+1} &= \underline{\mathrm{Y}}_{k+1} - \underline{\mathrm{Y}}_{k+1|k} \\ \mathbf{S}_{k+1} &= \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^{\mathrm{T}} + \mathbf{R}_{k+1} \\ \mathrm{K}_{k+1} &= \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^{\mathrm{T}}\mathbf{S}_{k+1}^{-1} \\ \underline{\mathrm{X}}_{k+1} &= \underline{\mathrm{X}}_{k+1|k} + \mathrm{K}_{k+1}\underline{\mathrm{Z}}_{k+1} \\ \mathbf{P}_{k+1} &= (\mathrm{I} - \mathrm{K}_{k+1}\mathbf{H}_{k+1})\mathbf{P}_{k+1|k} \end{aligned} \tag{B.5}$$

avec

- $\mathbf{F}_{k+1}$: jacobienne de la fonction **f**
- $\mathbf{H}_{k+1}$: jacobienne de la fonction **h**

La convergence de l'EKF n'est aucunement assurée car il s'agit d'une convergence locale. En fait, il existe de nombreux exemples pour lesquels la convergence du filtre dépend de l'initialisation de l'état à l'instant initial.

Annexe C

Probabilités et réseaux Bayésien

C.1 Langage de l'aléatoire

C.1.1 Phénomènes aléatoires

Un phénomène est dit aléatoire lorsqu'en recommençant une expérience dans des conditions aussi semblables que possible, le résultat de cette expérience est variable et ne peut pas être prédit de manière absolue et certaine. L'exemple le plus typique est celui du lancé de dé : à moins que le dé ne soit pipé, on ne sait pas à l'avance quelle face sera apparente. Une expérience retourne un résultat dit **aléatoire** lorsque le résultat ne peut pas être prédit. Cela peut être dû à notre incapacité à modéliser un système trop complexe comme la trajectoire exacte d'un dé sur une table.

C.1.2 Univers et réalisation

Dans l'exemple du dé à 6 faces, il y a six futurs possibles en fonction de la face visible du dé. Les futurs possibles sont désignés par la variable ω et portent différents noms : épreuve, résultat de l'expérience ou encore **réalisation du hasard** que nous privilégierons ici. L'ensemble de toutes les réalisations du hasard s'appelle **univers des possibles** (souvent abrégé en univers) et est noté Ω. Une expérience est modélisée par le choix de cet univers Ω. Dans le cas d'un jet de dé, l'univers classique associé est $\Omega = 1,2,3,4,5,6$. Les futurs possibles ont été caractérisés par la face visible mais auraient très bien pu l'être par le nombre de rebonds du dé ce qui aurait donc construit un autre univers. Par conséquent, l'univers est bien lié à la modélisation choisie pour une expérience.

C.1.3 Événements

Souvent, nous ne nous intéressons pas à une épreuve ω donnée mais à une partie de Ω qui sera appellée **événement**. Par exemple l'événement "le dé sort un nombre pair" correspond à la partie $2,4,6$ de l'univers $\Omega = 1,2,3,4,5,6$.

C.2 Calcul de probabilités

C.2.1 Définitions et premières propriétés

Une fois défini notre univers, l'objectif est de quantifier quelles sont les versions les plus probables, c'est à dire quelle probabilité associer, si possible, à chaque élément ω ou au moins à chaque événement A. Pour ce faire, une fonction P est définie sur l'ensemble des événements τ et vérifie à minima les propriétés suivantes :

— P est à valeur dans $[0,1]$

— l'univers Ω est de probabilité unité : $P(\Omega) = 1$.

— si A et B sont deux événements disjoints alors la probabilité associée à leur union est la somme de leurs probabilités : $P(A \cup B) = P(A) + P(B)$.

C.2.1.1 Définitions

Nous noterons :

— $\overline{A}$: l'événement contraire à A (qui correspond donc au complémentaire de A dans Ω).

— $A,B = A \cap B$: l'événement résultant de l'intersection de l'événement A et B : si on note C cet événement, pour avoir C, il est nécessaire que A et B soient réalisés

C.2.1.2 Propriétés

Soient A et B deux parties (ou événements) de Ω

— $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

— $P(\overline{A}) = 1 - P(A)$

— $P(\emptyset) = 0$

— si $A \subset B$ alors $P(A) \leq P(B)$

C.2.2 Probabilités conditionnelles

C.2.2.1 Formalisation

Soient deux événements A et B et P(A|B) la probabilité de l'événement A sachant l'événement B alors :

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{C.1}$$

C.2.3 Théorèmes

C.2.3.1 Formule des probabilités totales

Lorsqu'une expérience ne peut donner que des résultats A_1, A_2, ... en nombre fini ou dénombrable, c'est à dire si :

$$\Omega = \bigcup_k A_k$$

(l'union étant disjointe) alors tout événement B peut se décomposer sous la forme d'une union disjointe :

$$B = (B \cap A_1) \cup (B \cap A_2) \cup .. = \bigcup_k (B \cap A_k)$$

Alors la **formule des probabilités totales** donne :

$$P(B) = \sum_k P(B \cap A_k) = \sum_k P(A_k)P(B|A_k) \tag{C.2}$$

C.2.3.2 Théorème de Bayes

Soit deux événements A et B de probabilité non nulle alors

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{C.3}$$

Le théorème de Bayes est parfois appelé *formule de probabilités des causes (connaissant les conséquences)* puisqu'il permet de renverser le sens de la causalité entre les événements. C'est en particulier grâce à ce théorème qu'il va être possible pour nous de faire de inférence bayésienne : une conséquence est observée (la non détection d'une partie par exemple) qui peut potentiellement avoir de multiples causes (erreur du détecteur, mauvaise estimation courante etc) et il va être possible de calculer les probabilités à posteriori (sachant le résultat de l'expérience) afin de modifier nos différentes hypothèses.

C.2.3.3 Formule des probabilités composées

Supposons avoir des événements A_1, A_2, ..., A_n dont l'intersection est supposée de probabilité non nulle alors :

$$P(A_1, A_2) = P(A_1)P(A_2|A_1) \tag{C.4}$$

par simple définition d'une probabilité conditionnelle. De même, nous pouvons écrire :

$$P(A_1, A_2, A_3) = P(A_1, A_2)\, P(A_3|A_1, A_2) \tag{C.5}$$

$$= P(A_1)\, P(A_2|A_1)\, P(A_3|A_1, A_2) \tag{C.6}$$

$$\tag{C.7}$$

et par récurrence, nous aboutissons à :

$$P(A_1, A_2, ..., A_n) = P(A_1)\, P(A_2|A_1)\, P(A_3|A_1, A_2) \ldots P(A_n|A_1, ..., A_{n-1}) \tag{C.8}$$

C.3 Réseaux Bayésien

Un réseau bayésien est un modèle graphique probabiliste représentant des variables aléatoires sous la forme d'un graphe orienté acyclique voir la figure C.1 pour un exemple.

L'intérêt des réseaux bayésiens est multiple. Premièrement, ils permettent de représenter de manière "simple" les connaissances : les nœuds et les flèches indiquent rapidement quels événements sont pris en compte et de quels autres événements ils sont les causes ou conséquences. Il est donc beaucoup plus facile de discuter et de modifier une modélisation d'un système à l'aide de ces réseaux. De plus, comme nous l'avons déjà évoqué précédemment, ils permettent de faire de l'inférence bayésienne, c'est à dire de calculer les probabilités des données non observées (et potentiellement non observables ou très difficilement) en fonction des informations observées.

C.3.1 Un exemple simple

Un employé travaille sur des machines industrielles potentiellement dangereuses. Des accidents peuvent survenir et l'employé être blessé. Le risque d'accident est dépendant de la complexité de la machine et de l'expérience de l'employé : plus la machine est complexe, plus l'employé risque de faire une erreur et de se blesser ; plus il est expérimenté, mieux il connaît la machine et donc moins il a de chances de se blesser. "Expérience" et "Complexité" sont donc les deux facteurs déterminants du risque d'accident (voir figure C.1).

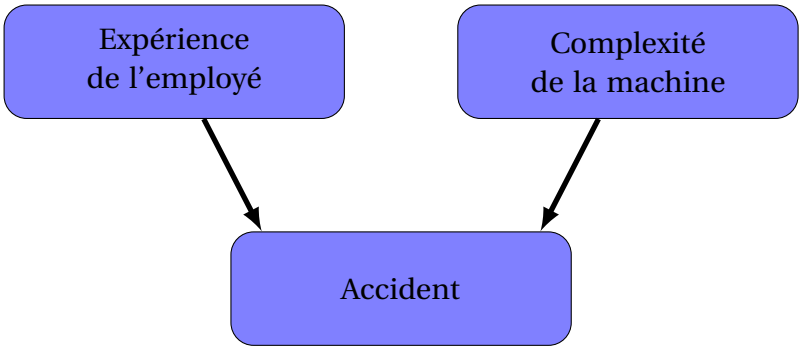


FIGURE C.1 – Exemple de réseau bayésien simple. Deux facteurs sont utilisés pour modéliser les causes d'accidents : l'expérience de l'employé et la complexité de la machine.

Il est évident qu'avec ces deux seuls facteurs, le modèle n'est pas déterministe : l'employé peut être expérimenté et la machine simple, un accident est toujours possible. D'autres facteurs peuvent influencer l'accident : l'employé peut être fatigué, la machine légèrement enrayée... L'accident est donc bel et bien aléatoire mais il est possible de le probabiliser en fonction des facteurs de risques identifiés.

TABLEAU C.1 – Probabilité d'accidents en fonction des facteurs de risques "Expérience" et "Complexité"

Complexité	faible		moyenne		élevé	
Expérience	novice	expert	novice	expert	novice	expert
accident	1%	0.4%	1.5%	0.5%	2.4%	0.7%
non accident	99%	99.6%	98.5%	99.5%	97.6%	99.3%

Un réseau bayésien permet d'intégrer des connaissances d'expert (comme les facteurs déterminants "Expérience" et "Complexité" dans l'exemple) et des données statistiques (les différentes valeurs pour remplir le tableau C.1 peuvent venir de statistiques réalisées dans l'entreprise).

C.3.2 Construction des réseaux bayésiens

Construire un réseau bayésien comporte donc deux aspects essentiels :

— définir le graphe du modèle : quels sont les différents événements pris en compte, lequel influence lequel ...

— définir les tables de probabilités de chaque variable (nœud) conditionnellement à ses causes (nœuds parent)

Le graphe est aussi appelé *structure* du modèle et les tables de probabilités conditionnelles ses *paramètres*. Le plus souvent la structure est définie par des "experts" du domaine et les paramètres sont issus de statistiques expérimentales.

C.3.3 Utilisation d'un réseau bayésien

En fonction des informations observées (évidence), la probabilité des données non observées est calculée. Ce phénomène est appelée inférence bayésienne. Par exemple, en fonction des symptômes d'un malade, les probabilités des différentes pathologies compatibles avec ces symptômes sont déterminées. Il est aussi possible de calculer la probabilité de symptômes non observés, et d'en déduire les examens complémentaires les plus intéressants.

C.3.4 Propriété de Markov générale

Un réseau bayésien vérifie la propriété de Markov générale. Si A, B, C sont des noeuds du réseau, que A est un parent de C tandis que B ne l'est pas alors, C et B sont indépendants conditionnellement à A, c'est à dire que :

$$P(C|A,B) = P(C|A)$$

C.4 Calcul des probabilités pour un réseau bayésien binaire de structure fixe

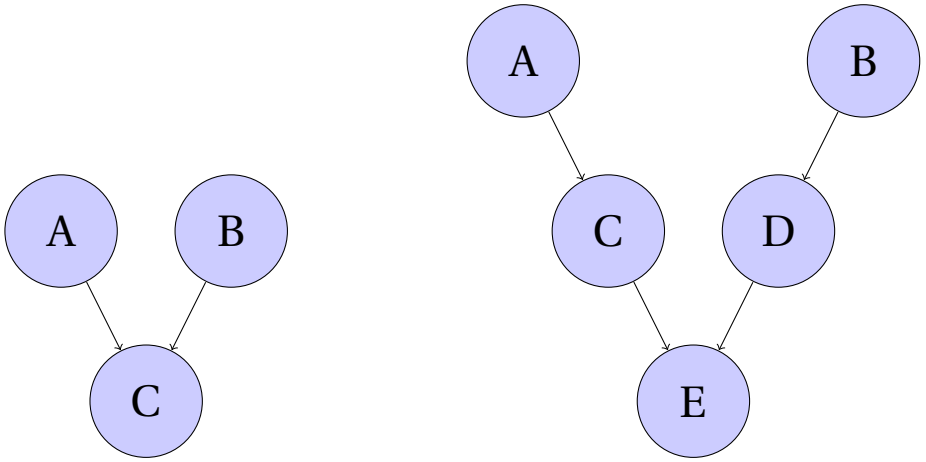
Pour calculer les probabilités dans les réseaux bayésiens, il existe des outils comme par exemple la bibliothèque Dlib [King, 2009] qui propose entre autres un outil graphique permettant de construire rapidement un réseau bayésien et de calculer les valeurs des différents nœuds en fonction des évidences indiquées. Bien que cet outil soit très pratique, il ne correspond pas tout à fait à nos besoins particuliers : nous avons un réseau dont la structure est fixe et dont tous les événements sont binaires. Il est possible avec ces hypothèses de calculer plus efficacement les probabilités, c'est ce que nous allons présenter ici.

Un réseau bayésien quelconque dont tous les événements sont binaires (ils peuvent uniquement prendre la valeur "vraie" ou "faux") est construit. Notons A_1, ..., A_n les différents événements constituant ce réseau. Nous allons construire une table $Table$ comme celle présentée dans le tableau C.2 : la table comportera donc $n+1$ colonnes (une colonne par nœud et une colonne pour les probabilités jointes) et 2^n lignes (pour créer toutes les combinaisons possibles "vrai", "faux des événements").

TABLEAU C.2 – Table complète des probabilités pour un réseau bayésien constitué des nœuds A_1, ..., A_n.

A_1	A_2	...	A_n	$P(A_1, A_2, ..., A_n)$
0	0	...	0	$P(A_1 = 0, A_2 = 0, \ldots, A_n = 0)$
0	0	...	1	$P(A_1 = 0, A_2 = 0, \ldots, A_n = 1)$
		...		...
1	1	...	1	$P(A_1 = 1, A_2 = 1, \ldots, A_n = 1)$

Pour calculer la valeur de la dernière colonne, nous utilisons la formule des probabilités composées (voir le § C.2.3.3) ainsi que les tables de probabilités conditionnelles du réseau bayésien.



(a) Réseau bayésien composé des événements A, B et C.

(b) Réseau bayésien composé des événements A, B, C, D et E.

FIGURE C.2 – Deux exemples simples de réseaux bayésiens.

Par exemple, pour le réseau bayésien de la figure C.2a, nous obtenons $P(A,B,C) = P(A)\,P(B)\,P(C|A,B)$. En fait, la formule des probabilités composées donnerait exactement $P(A,B,C) = P(A)\,P(B|A)\,P(C|A,B)$ mais comme A et B sont des événements indépendants, $P(B|A) = P(B)$. Avec, ceci, nous pouvons construire la table $Table_1$ présentée dans le tableau C.3.

TABLEAU C.3 – Table complète des probabilités $Table_1$ pour le le réseau bayésien de la figure C.2a

A	B	C	P(A, B, C)
0	0	0	$P(A=0,B=0,C=0)$
0	0	1	$P(A=0,B=0,C=1)$
0	1	0	$P(A=0,B=1,C=0)$
0	1	1	$P(A=0,B=1,C=1)$
1	0	0	$P(A=1,B=0,C=0)$
1	0	1	$P(A=1,B=0,C=1)$
1	1	0	$P(A=1,B=1,C=0)$
1	1	1	$P(A=C,B=1,C=1)$

Pour le réseau bayésien de la figure C.2b, la formule des probabilités composées donne :

$$P(A,B,C,D,E) = P(A)\,P(B|A)\,P(C|A,B)\,P(D|A,B,C)\,P(E|A,B,C,D) \tag{C.9}$$

En utilisant la propriété de Markov générale d'un réseau bayésien, la formule (C.9) peut être simplifiée :

$$P(A,B,C,D,E) = P(A)\,P(B)\,P(C|A)\,P(D|B)\,P(E|C,D) \tag{C.10}$$

Chacune des probabilités qui apparaît au dessus est alors connue par les tables de probabilités conditionnelles du réseau. Par conséquent, il est possible de calculer toutes

les valeurs $P(A_1 = 0, A_2 = 0, \ldots, A_n = 0)$, ..., $P(A_1 = 1, A_2 = 1, \ldots, A_n = 1)$.

Une fois cette table construite, pour calculer une probabilité donnée, il suffit de sommer les lignes correspondant à la situation cherchée :

$$\begin{aligned} P(A_i = 1) =& P(A_1 = 0, A_2 = 0, ..., A_i = 1, ..., A_n = 0) + P(A_1 = 0, A_2 = 0, ..., A_i = 1, ..., A_n = 1) \\ &+ ... + P(A_1 = 1, A_2 = 1, ..., A_i = 1, ..., A_n = 1) \\ =& \sum P(A_1 = \Phi, A_2 = \Phi, ..., A_i = 1, ..., A_n = \Phi) \text{ avec } \Phi \text{ pouvant prendre les valeurs 0 ou 1} \end{aligned}$$

Calculer une probabilité conditionnelle pourra se faire selon le même principe :

$$\begin{aligned} P(A_i = 1 | A_j = 1) =& \frac{P(A_i = 1, A_j = 1)}{P(A_j = 1)} \\ =& \frac{\sum P(A_1 = \Phi, A_2 = \Phi, ..., A_i = 1, ..., A_j = 1, ..., A_n = \Phi)}{\sum P(A_1 = \Phi, A_2 = \Phi, ..., A_j = 1, ..., A_n = \Phi)} \end{aligned}$$

Cela devient particulièrement efficace lorsque les probabilités à calculer sont toujours les mêmes : pour chaque probabilité à calculer, il est alors possible de sauvegarder les lignes de la table $Table$ utiles dans le calcul et il suffira dans la suite uniquement de faire une somme sur l'ensemble des lignes (ou deux sommes et une division pour les probabilités conditionnelles).

Ainsi, si pour le réseau bayésien de la figure C.2b, nous voulons calculer :

$$P(A = 1 | E = 1) = \frac{\sum P(A = 1, B = \Phi, C = \Phi, D = \Phi, E = 1)}{\sum P(A = \Phi, B = \Phi, C = \Phi, D = \Phi, E = 1)}$$

Alors parmi les 32 lignes de la table $Table_2$ présentée dans le tableau C.4, seules les lignes marquées par les flèches rouges nous intéresseront soit la moitié des lignes.

TABLEAU C.4 – Table complète des probabilités $Table_2$ pour le le réseau bayésien de la figure C.2b

A	B	C	D	E	P(A, B, C, D, E)	
0	0	0	0	0	P(A = 0, B = 0, C = 0, D = 0, E = 0)	
0	0	0	0	1	P(A = 0, B = 0, C = 0, D = 0, E = 1)	←
0	0	0	1	0	P(A = 0, B = 0, C = 0, D = 1, E = 0)	
0	0	0	1	1	P(A = 0, B = 0, C = 0, D = 1, E = 1)	←
0	0	1	0	0	P(A = 0, B = 0, C = 1, D = 0, E = 0)	
0	0	1	0	1	P(A = 0, B = 0, C = 1, D = 0, E = 1)	←
0	0	1	1	0	P(A = 0, B = 0, C = 1, D = 1, E = 0)	
0	0	1	1	1	P(A = 0, B = 0, C = 1, D = 1, E = 1)	←
0	1	0	0	0	P(A = 0, B = 1, C = 0, D = 0, E = 0)	
0	1	0	0	1	P(A = 0, B = 1, C = 0, D = 0, E = 1)	←
0	1	0	1	0	P(A = 0, B = 1, C = 0, D = 1, E = 0)	
0	1	0	1	1	P(A = 0, B = 1, C = 0, D = 1, E = 1)	←
0	1	1	0	0	P(A = 0, B = 1, C = 1, D = 0, E = 0)	
0	1	1	0	1	P(A = 0, B = 1, C = 1, D = 0, E = 1)	←
0	1	1	1	0	P(A = 0, B = 1, C = 1, D = 1, E = 0)	
0	1	1	1	1	P(A = 0, B = 1, C = 1, D = 1, E = 1)	←
1	0	0	0	0	P(A = 1, B = 0, C = 0, D = 0, E = 0)	
1	0	0	0	1	P(A = 1, B = 0, C = 0, D = 0, E = 1)	←
1	0	0	1	0	P(A = 1, B = 0, C = 0, D = 1, E = 0)	
1	0	0	1	1	P(A = 1, B = 0, C = 0, D = 1, E = 1)	←
1	0	1	0	0	P(A = 1, B = 0, C = 1, D = 0, E = 0)	
1	0	1	0	1	P(A = 1, B = 0, C = 1, D = 0, E = 1)	←
1	0	1	1	0	P(A = 1, B = 0, C = 1, D = 1, E = 0)	
1	0	1	1	1	P(A = 1, B = 0, C = 1, D = 1, E = 1)	←
1	1	0	0	0	P(A = 1, B = 1, C = 0, D = 0, E = 0)	
1	1	0	0	1	P(A = 1, B = 1, C = 0, D = 0, E = 1)	←
1	1	0	1	0	P(A = 1, B = 1, C = 0, D = 1, E = 0)	
1	1	0	1	1	P(A = 1, B = 1, C = 0, D = 1, E = 1)	←
1	1	1	0	0	P(A = 1, B = 1, C = 1, D = 0, E = 0)	
1	1	1	0	1	P(A = 1, B = 1, C = 1, D = 0, E = 1)	←
1	1	1	1	0	P(A = 1, B = 1, C = 1, D = 1, E = 0)	
1	1	1	1	1	P(A = 1, B = 1, C = 1, D = 1, E = 1)	←

Annexe D

Glossaire

amer Un amer est un point de repère fixe et identifiable sans ambiguïté utilisé pour la navigation maritime. Le terme amer est également utilisé en robotique mobile pour désigner les points de repère utilisés par un robot pour se repérer dans son environnement et calculer sa trajectoire. 48, 146

évidence Avoir une évidence pour un événement signifie connaître sa réalisation. Pour un événement binaire, il s'agit d'être capable de savoir s'il est vrai ou faux. Pour un autre type d'événement, il s'agit d'être capable de connaître sa valeur. Par exemple, si l'événement A correspond à la face visible du dé après un lancer, cet événement peut prendre les valeurs : $\{1,2,3,4,5,6\}$. Le dé est jeté, l'événement A devient une évidence : $A = 2$ si c'est la face numérotée 2 du dé qui est visible. 95, 102, 103, XIII

inférence Opération par laquelle une assertion considérée comme vraie rend vraie une autre assertion au moyen d'un système de règles. 31, 32

inférence bayésienne L'inférence bayésienne est une méthode d'inférence permettant de déduire la probabilité d'un événement à partir de celles d'autres événements déjà évalués. 76, 95, 99, 102, 103, XI, XIII

intégrité L'intégrité de l'estimation représente le fait que le vecteur de caractéristiques $\underline{X}$ à l'étape l est bien compris dans l'ellipse définie à l'aide de l'estimation $\underline{X}_{l|l}$ et de sa covariance associée $\mathbf{C}_{l|l}$. 46, 77

partie Une partie est une division, une portion, un morceau, une fraction d'un tout plus complexe, ici l'objet. Une partie peut ainsi être par exemple un œil, une bouche, ou un nez par rapport à un objet "visage". Une partie est censée présenter un caractère moins invariant que l'entité (objet par exemple) qui la comprend. 29

primitive Une primitive est un résultat renvoyé par un détecteur (un détecteur peut potentiellement retourner plusieurs primitives ou aucune). Une primitive peut ensuite éventuellement être associée à une partie. La partie correspond donc au concept tandis que la primitive associée est la réalisation. 29