Evitement d'obstacles par invariants visuels

Amaury Nègre

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : *Informatique*

préparée au laboratoire LIG et l'INRIA Rhône-Alpes, dans le cadre de l'École doctorale *Mathématiques, Sciences et Technologies de l'Information, Informatique*

présentée et soutenue publiquement par

Amaury NÈGRE

le 5 mars 2009

Titre :

Evitement d'obstacles par invariants visuels

Co-directeurs de thèse :

James CROWLEY et Christian LAUGIER

Composition du jury :

M.	Augustin LUX	Président
M.	Simon LACROIX	Rapporteur
M.	Didier AUBERT	Rapporteur
M.	Michel DHOME	Examinateur
M.	James L. CROWLEY	Directeur de thèse
M.	Christian LAUGIER	Co-directeur de thèse

Résumé

Notre travail se place dans le contexte de navigation visuelle en environnement ouvert et dynamique. Dans ce cadre, l'objectif est de faire évoluer un robot ou une voiture autonome dans un environnement composé d'objets pouvant interagir avec le robot, par exemple dans une rue en présence de véhicules, piétons, cyclistes, etc. Dans ce type d'environnement non conditionné, une des difficultés majeures est la détection des obstacles. Un autre point essentiel est de caractériser ces obstacles pour prédire leur comportement afin d'éviter ou de minimiser les chocs. En effet, tous les obstacles potentiels ne sont pas forcément dangereux dans l'immédiat. En outre, plusieurs contraintes se rajoutent dans le contexte de la robotique autonome, il est nécessaire de mettre en oeuvre des systèmes embarqués et temps réel avec des capteurs robustes et les moins chers possible.

Notre travail a consisté à mettre au point un détecteur visuel fondé sur des segments de crêtes dans l'espace d'échelle (gaussien et laplacien). Ce descripteur permet d'extraire des éléments saillants dans l'image, qui correspondent à des lignes d'intérêt naturel. Ces objets sont alors suivi d'une image à l'autre en temps réel afin de détecter les variations spatiales (correspondant à un déplacement dans l'image) ainsi que les variations d'échelle.

Afin de caractériser la dangerosité des obstacles, nous proposons de mesurer le temps avant collision (TTC) de chaque objet suivi. Cette mesure correspond à une distance dans le domaine temporel et est une information importante pour déterminer les obstacles les plus dangereux. Le TTC peut être évalué à partir du grossissement des objets dans l'image caméra, or une variation de taille correspondant à une variation d'échelle caractéristique, le TTC peut être calculé à partir de la trajectoire des objets dans l'espace d'échelle.

Pour valider notre méthode, nous avons effectué des tests avec un robot mobile équipé d'une caméra monoculaire dans un environnement urbain avec des objets fixes et mobiles. Les expériences montreront que la méthode est utilisable dans le cadre de la navigation en environnement extérieur et dynamique et devrait permettre de réaliser des applications d'évitement d'obstacles ou d'aide à la conduite.

Table des matières

Chapitre 1

Introduction

1.1 Aide à la conduite automobile

L'amélioration de la sécurité routière est – avec la réduction de la consommation en carburant – l'un des principaux enjeux de l'automobile moderne. On a ainsi pu voir se développer ces dernières années un certain nombre de dispositifs d'assistance à la conduite, avec notamment l'ABS, l'ESP, qui permet de contrôler et corriger les trajectoires dans les courbes, ou encore les systèmes de régulation Autocruise control (ACC) permettant de réguler automatiquement la vitesse pour garder une distance suffisante avec un véhicule suivi.

Dans ce contexte, la perception de l'environnement joue un rôle primordial. Les systèmes actuels privilégient des capteurs fournissant des informations de distances, comme des systèmes radars ou des détecteurs laser, du fait de la facilité à traiter les informations reçues. Cependant, on peut noter plusieurs points limitant l'utilisation de ces capteurs. D'une part, leur coût élevé ralentit grandement la distribution de ces produits pour le grand public et d'autre part ces capteurs sont assez mal adaptés à des environnements complexes tels que le milieu urbain où la multitude d'éléments dangereux impose un large champ de détection et une bonne capacité d'interprétation des données perçues. Les capteurs vidéos ont pour leur part été souvent négligés de part la complexité de traitement exigeant des ressources importantes de calculs. Toutefois, l'évolution permanente des processeurs embarqués devrait permettre d'utiliser ces capteurs intéressant pour leur faible coût et la quantité d'informations qu'ils fournissent. C'est dans ce contexte que nous situerons nos travaux.

1.2 Problème et objectifs

Dans cette étude, nous nous intéresserons au problème de la détection visuelle d'obstacles dans un environnement urbain. Dans ce type d'environnement, la diversité et la dynamique des éléments compliquent sérieusement cette tâche. Il est en effet nécessaire de percevoir tous les types d'obstacles ou de dangers potentiels, comme des voitures, des piétons, des cyclistes, etc. L'aspect dynamique joue aussi un rôle important, les obstacles mobiles ne sont pas à considérer de la même manière que les éléments fixes. Il est évidant qu'un obstacle qui se rapproche représente plus de risque que des éléments fixes ou qui s'éloignent du véhicule.

Pour répondre à ce problème, nous nous sommes fixé trois principaux objectifs :

- La détection des obstacles devra être le plus générique possible et adaptée aux éléments courant de l'environnent urbain.
- Il sera nécessaire d'estimer les risques des différents obstacles pour naviguer de manière sûre dans cet environnement. Cette estimation des risques devra tenir compte des aspects dynamiques du véhicule ainsi que des obstacles mobiles.
- Les temps de calculs devrons être suffisament courts pour permettre une utilisation temps réel (c'est-à-dire en ligne).

1.3 Solution proposée

Le problème de la détection d'obstacle a été traité de différentes manières en utilisant des capteurs variés, tels que des capteurs lasers, radar ou visuels. Nous nous somme orienté sur une approche vision car les capteurs caméras sont les capteurs les moins coûteux et généralement les moins encombrant, ce sont donc les plus faciles à intégrer dans des véhicules destinés au grand public. Dans ce manuscrit, nous allons ainsi proposer une méthode de détection des obstacles fondée uniquement sur des informations visuelles. Cette méthode pourra être décomposée en trois étapes que nous allons décrire plus précisément.

La première étape consistera à extraire des éléments correspondant à des zones d'intérêt dans une image. Cette extraction devra être bien adaptée aux obstacles et aux objets présents dans un environnement urbain. Afin d'être robuste aux différentes transformations dues aux mouvements de la caméra, mais aussi aux variations des conditions extérieures telles que l'éclairage et le mouvement des objets dynamiques, nous utiliserons des invariants visuels fondés sur la notion d'espace d'échelle.

La deuxième étape de calcul consistera à suivre l'évolution des différents éléments détectés afin de pouvoir estimer la dynamique des obstacles. La méthode proposée sera fondée sur un filtre bayésien appelé filtre à particules, de manière à être adaptée au modèle dynamique des objets et au modèle d'observation.

Enfin, la troisième étape permettra de déterminer les risques de collision associés à chaque objet suivi. Pour cela nous avons choisi d'utiliser la notion de temps avant collision qui est une distance temporelle entre l'observateur et les obstacles. De plus, nous verrons que cette mesure est particulièrement adaptée à un capteur visuel et s'intègre bien avec l'étape précédente.

1.4 Plan de lecture

Le chapitre 2 constitue une étude bibliographique du problème de la détection et de l'évitement d'obstacles. Ce chapitre présentera les différentes catégories d'obstacles et les solutions adaptées à chacune de ces catégories. Nous étudierons notamment les algorithmes de détections d'obstacles spécialisés permettant de détecter uniquement un certain type d'objets, puis nous verrons différentes méthodes plus générales qui permettent de détecter des obstacles plus variés, ce qui est de fait plus intéressant dans un environnement incontrôlé. Ensuite, ce chapitre présentera plusieurs éléments permettant de caractériser la dangerosité des obstacles dans le cadre de la conduite automobile.

Le chapitre 3 présentera en détail notre méthode de détection d'obstacles ou d'amers naturels dans une image fixe. Tout d'abord, ce chapitre introduira la notion d'espace d'échelle en développant les aspects théoriques et pratiques. La représentation multi-échelle d'une image consiste à introduire un paramètre d'échelle jouant le rôle de niveau de détail. En utilisant les dérivées secondes de cet espace, nous verrons qu'il est possible d'extraire une échelle caractéristique pour chaque pixel de l'image correspondant à la taille caractéristique des éléments présents dans l'image. Cette échelle est importante car elle permet d'adapter le détecteur en fonction de la taille des objets et d'être donc invariant au changement d'échelle. Ayant posé les bases théoriques de l'espace d'échelle, nous nous intéresserons alors à l'extraction des éléments importants dans une scène observée par une caméra. Les points d'intérêts tels que les extrema dans l'espace d'échelle Laplacien constituent un premier élément de réponse, mais une étude plus approfondie soulèvera un certain nombre de limites notamment lorsque les objets présentent des formes allongées. Ce type d'objet étant courant dans un environnement urbain, nous allons donc étendre la notion de point d'intérêt pour détecter des lignes

d'intérêts et plus précisément des segments de crête.

Le chapitre 4 s'intéresse au suivi dynamique des éléments détectés. Ce chapitre présentera tout d'abord une étude des différentes méthodes bayésiennes de filtrage permettant d'estimer l'état d'un système dynamique au cours du temps, tels que le filtre de Kalman et le filtre particulaire. Ensuite, le chapitre présentera en détail la méthode de filtrage choisie et en particulier les modèles dynamiques et les modèles d'observations propres à notre problème de suivi de segments de crêtes dans l'espace image. Pour le modèle d'observation, nous étudierons deux types d'informations : le premier modèle, utilisant la fonction de détection des segments de crêtes permettra de maintenir les éléments sur les zones d'intérêts, et le second modèle prenant en compte l'apparence évitera les erreurs d'associations. Ces deux modèles seront alors fusionnés pour utiliser les deux informations. Enfin, nous présenterons une méthode de suivi multi-cibles permettant de suivre en permanence un nombre constant de cibles en réinitialisant automatiquement les cibles perdues ou ayant fusionné avec une autre cible.

Le chapitre 5 sera consacré à l'évaluation des risques de collision. Nous verrons que le temps avant collision (TTC) est une mesure pertinente pour la caractérisation du danger puisque le TTC représente le temps qu'il reste avant d'entrer en contact avec un obstacle, un TTC faible caractérisant ainsi un risque élevé. De plus l'intérêt du TTC réside dans le fait qu'il peut être calculé par des méthodes purement visuelles, sans avoir à connaître la distance des obstacles. Pour cette raison, cette mesure est utilisée par de nombreux êtres vivants sous la forme d'un réflexe appelé *looming* permettant d'éviter des objets se rapprochant très vite de l'observateur. Après avoir défini le modèle de caméra utilisé, nous verrons dans ce chapitre qu'il est possible d'estimer ce TTC pour tous les éléments suivis dans l'image à partir de leur échelle (au sens de l'espace d'échelle) et de la variation de cette échelle. Ensuite, nous proposerons une méthode pour estimer ce TTC en pratique, en utilisant des données d'échelle pouvant être bruitées. Cela s'effectuera en utilisant un filtre des moindres carrés récursifs qui permet de filtrer des données bruitées tout en calculant le TTC.

Dans le chapitre 6 dédié aux expérimentations et à l'évaluation des algorithmes présentés dans les trois chapitres précédents, nous présenterons tout d'abord l'architecture matérielle et logicielle de la plateforme utilisée. Nous décrirons également les solutions logicielles que nous avons développées pour mener les expérimentations, ainsi qu'une implantation sur processeur graphique (GPU) qui permettra d'obtenir des temps de calculs les plus faibles possible en utilisant aux mieux l'architecture parallèle des GPU. Ensuite, nous évaluerons séparément les différentes parties du processus de détections d'obstacles. Nous étudierons dans un premier temps la répétabilité

du détecteur de segments de crête proposé au chapitre 3 qui permettra de valider la stabilité du détecteur par rapport aux différentes transformations de l'image. Dans un second temps nous évaluerons notre algorithme de suivi sur une scène statique puis sur une scène dynamique. Enfin, pour évaluer l'ensemble du processus, nous distinguerons deux types d'applications. Dans la première, un seul obstacle sera sélectionné à la main et nous montrerons que le TTC estimé pour cet obstacle permet au robot de s'arrêter avant la collision. Dans la seconde application, la totalité des éléments suivis dans l'image seront utilisés pour effectuer une tâche de navigation réactive (c'est-à-dire sans planification) avec évitement d'obstacles.

Chapitre 2

Navigation visuelle en environnement extérieur et dynamique

2.1 La détection d'obstacles

La détection d'obstacles joue un rôle fondamental dans le contexte des véhicules intelligents. Que ce soit en conduite automatique ou en assistance à la conduite, un des principaux problèmes est la perception et l'évaluation des zones à risques. Cependant, suivant les applications, la notion d'obstacle peut varier et donc les méthodes de détections peuvent être de nature différentes.

Comme le montrent Bertozzi *et al.* dans [BBF00] et [BBC+02], on peut classifier les algorithmes de détection d'obstacles en deux catégories : les algorithmes spécialisés dans la reconnaissance d'obstacles spécifiques (par exemple des piétons, des voitures, des cyclistes, etc...), et les algorithmes de détections d'obstacles génériques.

2.1.1 Algorithmes de détections spécialisés

Les algorithmes de détections d'obstacles spécialisés consistent généralement à analyser une image simple et à extraire les objets à l'aide d'indices divers comme les symétries, les ombres, la texture ou la répartition des couleurs. Parmi les détecteurs de véhicules, Marmoiton *et al.*[MCD00] utilisent les feux arrières des voitures pour la détection et le suivi. En constatant que les voitures ont un axe de symétrie vertical Broggi *et al.*[BCA04] ont mis au point un détecteur de symétrie multi-échelle permettant d'extraire les voitures quelle que soit leur distance par rapport

au véhicule. Dans [HDS04] un détecteur de symétrie similaire est fusionné avec un détecteur de lignes sombres afin d'améliorer la détection en tenant compte du fait que les voitures projettent toujours une zone d'ombre sur la route. D'autres méthodes utilisent des modèles géométriques déformables qui sont ajustés sur l'image pour pouvoir détecter la configuration du véhicule [FWSB95, CHDLEA04] (voir Figure 2.1).

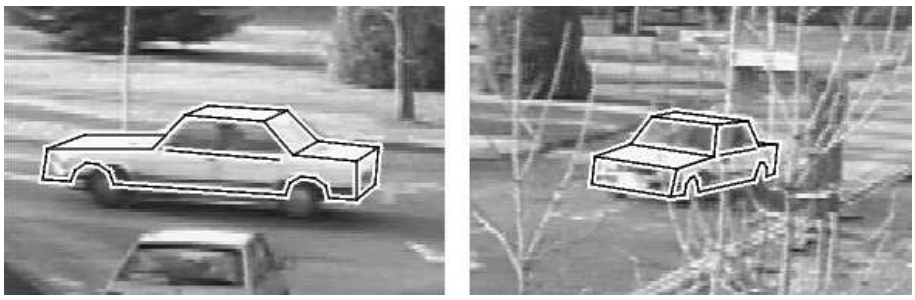


FIG. 2.1 – Détection et suivi de véhicules avec des modèles déformables [FWSB95]

Pour détecter des piétons, il existe des caméras infrarouges permettant d'observer la chaleur émise par les corps humains, dans ce cas les piétons correspondent à des zones de forte intensité dans l'image, il est alors relativement simple de les détecter par seuillage de l'intensité lumineuse et en utilisant des critères sur la symétrie, la forme ou la taille des boîtes englobantes pour les distinguer des autres corps chauds [ND02, BBG+03, XLF05, BFC+04]. Dans des images de caméras classiques, la détection est plus complexe du fait des nombreuses variations de l'apparence dues entre autres aux changements de postures et la diversité des vêtements portés. Pour ces raisons, il est difficile de spécifier intuitivement un modèle, il est donc nécessaire d'utiliser des méthodes d'apprentissage pour construire automatiquement ces modèles. Un des détecteurs les plus connus est celui de Viola, Jones et Snow [VJS05] (voir Figure 2.2) fondé sur le classifieur Adaboost en cascade [VJ01] en utilisant comme caractéristique des différences des pixels entre l'image et des motifs binaires qui sont appris automatiquement. D'autres détecteurs utilisent des réseaux de neurones pour l'apprentissage [ZT00] ou des machines à vecteurs de support [PP99], les caractéristiques visuelles servant à la détection étant la disparité (caméra stéréo) pour le premier article, les coefficients d'ondelettes pour le second, ou encore les contours dans [Gav00].

Ces détecteurs spécialisés sont utiles lorsque les scènes sont peu complexes, comme sur une autoroute où les seuls obstacles sur la route sont des véhicules ou bien lorsque l'application n'a qu'une seule tâche précise,

FIG. 2.2 – Exemple de détection de pietons avec le détecteur de Viola, Jones et Snow [VJS05]

comme surveiller et éviter les piétons, toutefois, dans des contextes plus complexes, les obstacles ne se limitent pas à des véhicules et des piétons, il est donc nécessaire d'utiliser des algorithmes de détection d'obstacles génériques.

2.1.2 Détection d'obstacles génériques

Les méthodes de détection d'obstacles génériques se fondent sur une définition beaucoup plus large des obstacles. Bertozzi *et al.* dans [BBC+02] considèrent que détecter les obstacles revient à identifier les zones libres, c'est-à-dire qu'un obstacle est défini comme un objet qui peut entraver la trajectoire du véhicule. Ainsi tout objet qui dépasse du sol ou de la route, peut être considéré comme un obstacle. Bien que cette définition paraisse très simple, les techniques utilisées peuvent s'avérer complexes et de nature très diverses.

Détection par télémètre laser

Pour des raisons de précision et de simplicité, le télémètre laser [EW00] est très utilisé pour la détection d'obstacles. Ce type de capteur permet de calculer la distance de tout objet réfléchissant dans l'axe d'un faisceau laser en mesurant le temps de vol de la lumière. Le faisceau laser peut être réfléchi par un miroir pivotant permettant alors de scanner tout un plan. La résolution des télémètres standard est de l'ordre du millimètre en distance et du dixième de degré en angle pour une précision effective de l'ordre de 5 à 10 mm. Avec un tel capteur, il est alors possible de localiser précisément

tout objet physique, et donc d'extraire tous les éléments situés au dessus du niveau du sol qui sont alors identifiés à des obstacles.

Néanmoins, plusieurs contraintes limitent l'utilisation de tels capteurs : la première est une contrainte de coût, en effet le prix très élevé de ces capteurs limite leur utilisation à grand échelle. La seconde contrainte est une contrainte de champ de vue. La zone de détection correspondant généralement à un demi-plan, ces capteurs doivent être placés près du sol et de manière parfaitement horizontale pour détecter tout types d'obstacles. Cependant, dès que le sol n'est pas plat ou lorsque la voiture est inclinée (lors d'un virage ou d'un freinage), le sol peut être détecté comme un obstacle, ou bien des obstacles peuvent sortir du champ de détection. Pour ces raisons des méthodes visuelles (c'est-à-dire effectuant un traitement de données de caméras embarquées) peuvent être plus intéressantes car la zone d'observation couverte par une caméra est généralement très large.

Détection par le mouvement

Il existe de nombreuses techniques pour détecter des objets mobiles avec une caméra fixe, ces méthodes sont souvent utilisées pour la détection des obstacles par l'infrastructure. Ces techniques utilisent une caméra classique noir et blanc, ou couleur, et consistent à faire une soustraction de fond, c'est-à-dire à faire une différence entre la couleur courante d'un pixel et la couleur de l'arrière plan, préalablement apprise, les pixels ayant une forte différence correspondent donc aux objets mobiles (les objets fixes sont utilisés pour apprendre la couleur de fond et ne sont donc pas détectés). Des modèles de fond plus ou moins complexes peuvent être utilisés pour diminuer la sensibilité aux ombres, aux changements d'illumination et au bruit des caméras [ED00, JSS02, MP04]. Plusieurs caméras fixes peuvent être utilisées simultanément en fusionnant les données afin d'augmenter la confiance et le champ de vue [CLFK01]. Il est aussi possible d'utiliser une caméra stéréo et utiliser l'image de disparité pour distinguer les objets de l'arrière plan et les obstacles [EKB98].

Toutefois, ces solutions ne conviennent pas lorsque les caméras sont embarquées dans des véhicules mobiles, l'arrière plan changeant en permanence. Dans [EKB98], le modèle de l'arrière plan autorise des mouvements de caméra lents, mais avec une caméra monoculaire, cela devient quasiment impossible lorsque la caméra se déplace rapidement. Ainsi, pour les caméras embarquées, il est nécessaire d'utiliser une autre type d'information que la variation de couleur ou d'intensité lumineuse. Si on utilise le flux optique (le mouvement des points dans l'image), il est par contre possible de détecter des objets mobiles en évaluant le mouvement propre de la caméra

entre deux images, puis, en utilisant la contrainte épipolaire, il est possible de faire ressortir tous les points de l'image qui ne se déplacent pas de la même manière que l'arrière plan [ASB94, Tor95] (voir Figure 2.3). Cependant, ces méthodes sont assez coûteuses et nécessitent un calcul de flux optique dense et précis [HS81, LK81, BWF+05], ce qui est difficile à obtenir en temps réel.

FIG. 2.3 – Segmentation du mouvement utilisant la méthode de P. Torr [Tor95], dans cette séquence la caméra et le camion bougent, ce qui permet d'extraire deux mouvements apparents : le fond et le camion et donc de détecter les obstacles mobiles.

Détection du plan du sol

Parmi les derniers types de détection d'obstacles visuels, on peut noter les méthodes fondées sur la détection du plan du sol. Ainsi, au lieu d'identifier les obstacles, on peut procéder de manière inverse en évaluant les zones libres qui correspondent au sol et sont donc non encombrées et non dangereuses. On peut classer ces méthodes en deux catégories : celles fondées sur le flux optique et celles utilisant des images stéréoscopiques. Les méthodes utilisant le flux optique consistent à estimer un modèle de déplacement des points du sol dans l'image, à partir du mouvement propre du robot (qui peut être obtenu par les capteurs du robot ou bien en estimant le flux optique global). Lorsque l'on connaît ce modèle du sol, il est alors facile de déterminer les points de l'image qui suivent ce modèle et les autres, qui correspondent aux obstacles [Kru99, KK03]. De la même manière, de nombreuses méthodes utilisent une caméra stéréo pour extraire le plan du sol [LAT02, SO06]. Les points situés au dessus de ce plan peuvent alors être considérés comme des

obstacles. Afin de combiner les points forts des deux méthodes, Braillon [Bra07] a fussioné les deux méthodes de détection (flux optique et stéréo) en utilisant une grille d'occupation de l'environnement.

2.2 Estimation des risques de collision

Dans un environnement urbain, un véhicule est en général entouré d'obstacles de tout genre qui peuvent être fixes ou mobiles, et représentant un danger plus ou moins imminent.

2.2.1 Distance aux obstacles

Tout d'abord, une des informations importantes pour prédire la collision est la distance aux obstacles, il est en effet évident qu'un obstacle lointain est moins dangereux qu'un obstacle proche. Cette information peut être obtenue directement par des capteurs de distance (télémètre laser, radar, ou caméra stéréo), mais avec une caméra monoculaire, la distance est plus compliquée à mesurer. Dans [SMS03], la distance des véhicules est calculée en déterminant le point de contact des objets détectés avec le sol dans l'image et en reprojetant ce point sur le plan du sol. Cette méthode suppose donc que le sol est parfaitement plat et nécessite en plus de calibrer parfaitement la caméra utilisée (position par rapport au sol et paramètres intrinsèques de la caméra). Dans des situations plus complexes (sol en pente ou non planaire) ou lorsque les obstacles détectés ont une hauteur inconnue, il est impossible de mesurer directement la distance. Par ailleurs, lorsque les objets sont mobiles, la distance seule ne permet pas d'exprimer la dangerosité des obstacles. Un obstacle éloigné mais approchant rapidement peut en effet être plus dangereux qu'un obstacle proche mais qui s'éloigne. Pour cette raison, il est nécessaire de déterminer d'autres critères plus adaptés à un capteur visuel et prenant en compte l'aspect dynamique de l'environnement.

2.2.2 Critères dynamiques

En conduite automobile, la notion de distance de sécurité constitue un critère important pour la sécurité de la conduite. Cette distance est définie comme la distance minimum permettant au véhicule conduit de s'arrêter avant d'entrer en contact avec le véhicule qui nous précède si celui-ci stoppe net. En fait cette distance dépend uniquement de la vitesse du véhicule piloté, pour un véhicule classique conduit par un humain, on estime que la distance de sécurité correspond à la distance parcourue par le véhicule

pendant un délai d'au moins deux secondes. La distance de sécurité d_s peut ainsi s'exprimer par la formule suivante :

$$d_s = k \cdot v$$

où k est une constante définie pour le véhicule et v la vitesse du véhicule. Dans [SFN+04] la distance temporelle $t = \frac{d}{v}$ entre le véhicule contrôlé (avançant à la vitesse v) et un véhicule observé (à la distance d) est ainsi évaluée pour contrôler la vitesse du véhicule. Toutefois, ce critère ne tient pas compte de la vitesse de l'obstacle considéré et nécessite de connaître sa distance et la vitesse du véhicule commandé.

Pour s'affranchir du calcul de distance, un autre critère – utilisable avec un simple capteur visuel – est nécessaire. Introduit par Lee [Lee76] dans les années 70, le temps avant collision est une mesure temporelle des obstacles. Bien que pouvant être défini comme le rapport entre la distance observateur/obstacle et la vitesse relative de rapprochement, cette mesure peut être évaluée avec des capteurs visuels, ce qui explique pourquoi ce critère est utilisé dans la nature par un certain nombre d'animaux [Lee81, LYR92]. Le réflexe appelé *looming* chez l'être humain et certains animaux utilisent en effet le grossissement des objets dans l'espace visuel pour détecter les obstacles dangereux approchant rapidement [SCG62, RB78, SF98].

2.3 Méthode proposée

Dans ce manuscrit, nous allons proposer une nouvelle approche de détection et de caractérisation des obstacles. Nous allons tout d'abord étudier une méthode de détection d'obstacles potentiels dans une image statique à partir de caractéristiques géométriques et différentielles, telle que la forme et le contraste. Ces obstacles devrons pouvoir être détectés de loin comme de près, c'est pourquoi le détecteur doit être invariant au changement d'échelle, pour permettre cela, nous allons utiliser la notion d'espace d'échelle, défini par Witkin [Wit83], qui permet de représenter une image et de définir des opérateurs à plusieurs niveaux de détails. On pourra noter que, même si ce détecteur est bien adapté aux objets allongés et contrastés, ce détecteur peut être considéré comme un détecteur d'obstacles génériques, car la détection ne se limite pas à une catégorie d'obstacles bien définie [NCL08].

Une fois ces premières pistes d'obstacles détectées, il sera nécessaire de suivre ces objets au cours du temps afin de déterminer leur trajectoire dans l'espace image, ou dans l'espace monde. Le déplacement et la vitesse de ces objets vont alors permettre de distinguer les cibles dangereuses.

Pour évaluer cette dangerosité, nous avons mis au point une méthode d'évaluation du temps avant collision fondée sur le changement d'échelle des cibles suivies. En effet, lorsque un objet s'approche de la caméra, on peut constater que sa taille augmente dans l'image (de manière non linéaire). Cela est équivalent à un décalage en échelle de cet objet dans l'espace d'échelle [NBCL06]. Ayant effectué le suivi des points non seulement en position mais aussi en échelle, il sera alors possible d'utiliser cette dernière composante pour mesurer le temps avant collision et donc obtenir un indice de dangerosité des cibles suivies.

Chapitre 3

Détection des obstacles et amers naturels dans une image fixe

3.1 Introduction

En observant une scène capturée par une caméra dans un environnement extérieur de type urbain, nous pouvons faire ressortir plusieurs critères qui vont permettre de mettre au point un détecteur capable d'extraire les éléments courants de cet environnement :

- Les objets (voiture, pietons, lignes blanches, murs, etc.) correspondent à des zones plus ou moins uniformes sur un fond de couleur différente ;
- La plupart de ces objets (ou partie d'objets) ont soit une géométrie circulaire ou elliptique (voiture, phare, visages, etc.) soit une forme linéïque (piéton éloigné, poteaux, lignes de marquage, etc.) ;
- La taille des objets dans l'image dépend non seulement de leur taille réelle mais aussi de la distance à laquelle ils se trouvent ;

D'après ces remarques, notre détecteur devra être sensible aux objets allongés ayant un fort contraste avec l'arrière plan. De plus ce détecteur devra être invariant au changement de taille, c'est-à-dire si un objet est détecté à une certaine taille, il doit l'être aussi à une taille différente (dans les limites de résolution de la caméra). Ce problème n'est pas trivial car lorsque un objet se rapproche, il peut apparaître certains détails qui peuvent modifier considérablement la texture, la couleur et l'apparence de l'objet. Par exemple si l'on considère un objet texturé avec des rayures blanches et noires, de loin on ne verra qu'une zone uniforme grise, alors que de près on verra distinctement les rayures, mais aucune zone uniforme grise. Afin de permettre

FIG. 3.1 – Scène observée par une caméra embarquée dans un véhicule en millieu urbain.

l'invariance à ce changement de taille, nous allons travailler dans l'espace d'échelle qui consiste à représenter l'image à différent niveaux de détails. Nous allons donc tout d'abord définir ce qu'est un espace d'échelles et les propriétés qui en découlent, puis nous verrons comment définir un opérateur dans cet espace d'échelle permettant de détecter les éléments importants de la scène.

3.2 Utilisation de l'espace d'échelle

3.2.1 Définition de l'espace d'échelle

L'espace d'échelle est une représentation multi-résolution d'un signal. Un tel espace permet de visualiser l'image originale, avec tous ses détails, et de pouvoir supprimer progressivement les détails, comme si l'on s'éloignait de plus en plus de l'image. Cet aspect est très important en vision et particulièrement pour la robotique mobile, car si un robot s'approche d'une zone, l'image qu'il a de la scène gagne progressivement en détails et l'image de la scène peut être considérablement modifiée, mais avec la représentation multi-échelle, l'image de la scène à un instant passé sera très proche de l'image courante à un certain niveau d'échelle.

De manière mathématique, l'espace d'échelle est défini comme l'ensemble des convolutions du signal par un filtre passe bas paramétrisé par la taille

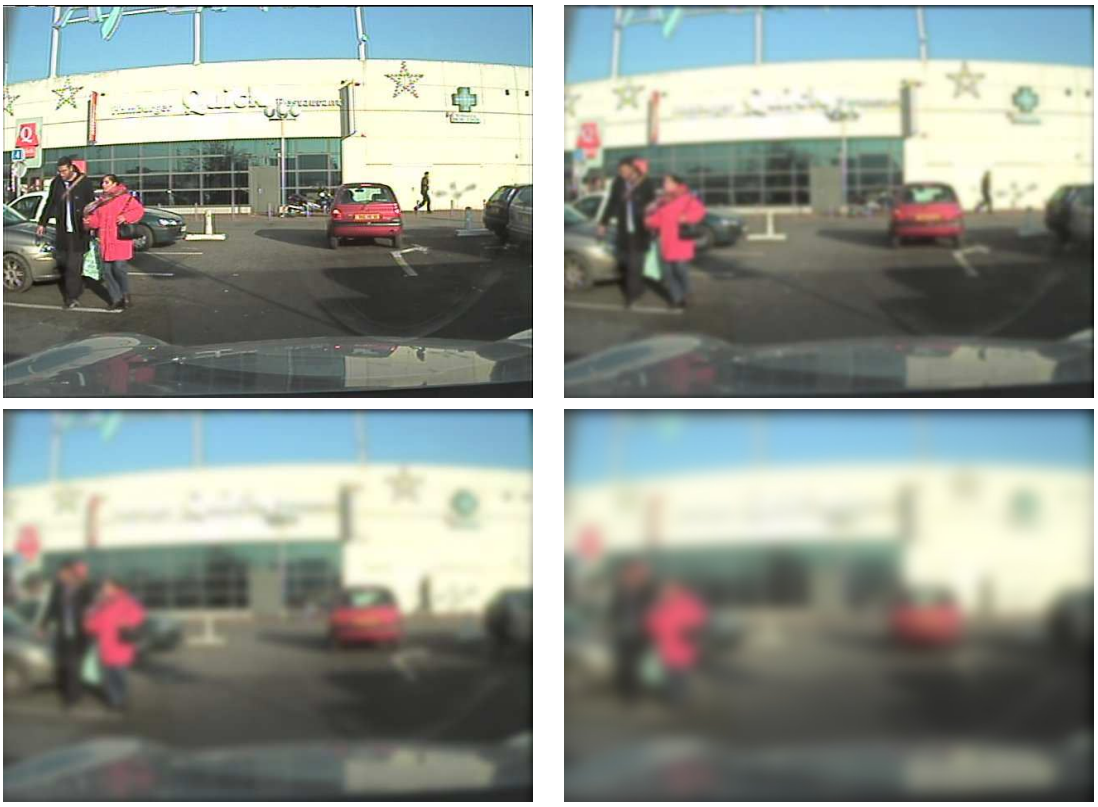


FIG. 3.2 – Représentation multi-échelle d'une image, les petits détails disparaissent progressivement lorsqu'on augmente l'échelle. A grande échelle, seul des éléments les plus larges sont visibles, c'est ce qui se passe lorsqu'on observe une scène de plus en plus loin.

caractéristique du noyau de lissage [Koe84, BWBD85, Lin94]. Cette taille caractéristique, représentée par la variable σ, correspond à la taille des éléments qui restent visibles dans l'image après application du filtre passe bas. Lorsque l'on augmente ce paramètre, on supprime ainsi progressivement le niveaux de détail.

Définition 1 *On considère une image I représentée par la fonction $f(x, y)$ qui associe à chaque pixel (x, y) son intensité. L'espace d'échelle linéaire (ou gaussien) associé à cette image est l'ensemble des images $L(x, y, \sigma)$ définies par la convolution de $f(x, y)$ avec un noyau gaussien d'écart type sigma :*

$$g_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3.1}$$

$$L(x, y, \sigma) = g_\sigma(x, y) * f(x, y) \tag{3.2}$$

où l'opérateur $*$ représente le produit de convolution :

$$(h_1 * h_2)(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h_1(x - u, y - v) \cdot h_2(u, v)\, du\, dv$$

Il est possible d'utiliser un filtres passe bas différent du noyau gaussien, toutefois le choix de ce noyau est intéressant car c'est le seul à vérifier les axiomes suivant (on note $L(x, y, \sigma) = (T_\sigma f)(x, y) = g_\sigma(x, y) * f(x, y)$) :

- Linéarité :
$$T_\sigma(af + bh) = aT_\sigma f + bT_\sigma h \tag{3.3}$$
- Invariance à la translation :
$$T_\sigma S_{(\Delta_x, \Delta_y)} f = S_{(\Delta_x, \Delta_y)} T_\sigma f \tag{3.4}$$
où $S_{(\Delta_x, \Delta_y)}$ est l'opérateur de translation : $(S_{(\Delta_x, \Delta_y)} f)(x, y) = f(x - \Delta_x, y - \Delta_y)$
- Structure de semi-groupe (en faisant le changement de variable $t = \sigma^2$) :
$$g_{t_1}(x, y) * g_{t_2}(x, y) = g_{t_1 + t_2}(x, y) \tag{3.5}$$
- Non création d'extrema locaux dans une direction.
- Non-augmentation des extrema locaux :
$$\partial_\sigma L(x, y, \sigma) \leq 0 \tag{3.6}$$
pour les maxima spatiaux et
$$\partial_\sigma L(x, y, \sigma) \geq 0 \tag{3.7}$$
pour les minimas spatiaux.
- Invariance aux rotations :
$$\forall \theta \in \mathbb{R}, g_\sigma(x, y) = g_\sigma(\cos\theta \cdot x + \sin\theta \cdot y, -\sin\theta \cdot x + \cos\theta \cdot y) \tag{3.8}$$
- Invariance au changement d'échelle[1] :
$$\exists \varphi, h / \hat{g_\sigma}(\omega_x, \omega_y) = \hat{h}(\frac{\omega_x}{\varphi(\sigma)}, \frac{\omega_y}{\varphi(\sigma)}) \tag{3.9}$$
Où $\hat{g}$ est la transformée de Fourier de g.
- Positivité :
$$g_\sigma(x, y) \geq 0 \tag{3.10}$$

[1]Les symboles "∃.../" signifient "il existe ... tel que"

– Normalisation :

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g_\sigma(x,y)\,dx\,dy = 1 \tag{3.11}$$

De manière équivalente l'espace d'échelle gaussien peut être défini comme la solution de l'équation de diffusion (paramétrisée en σ au lieu du paramètre courant $t = \sigma^2$) :

$$\frac{\partial L}{\partial_\sigma} = \sigma\nabla^2 L \tag{3.12}$$

avec les conditions initiales $L(x, y, 0) = f(x, y)$.

Ces propriétés sont intéressantes car l'espace d'échelle ainsi définit est invariant aux différentes transformations que sont les rotations, les translations et les changements d'échelles (c'est à dire que si une de ces transformations est appliquée à l'image, on retrouve la même transformationa à chaque niveau de l'espace d'échelle).

3.2.2 Dérivées de l'espace d'échelle

Définition 2 *On note $L_{x^m y^n}$ la dérivée de l'espace d'échelle L, d'ordre m en x et d'ordre n en y, défini de telle sorte que :*

$$\forall\sigma \in \mathbb{R}^+ : L_{x^m y^n}(x, y, \sigma) = \frac{\partial^{m+n}}{\partial x^m \partial y^n}(L(x, y, \sigma)) \tag{3.13}$$

Propriété 1 *Du fait de la commutativité de l'opérateur dérivée et de la convolution par une gaussienne, les dérivées de l'espace d'échelle linéaire peuvent être calculées directement par convolution avec des dérivées de gaussiennes :*

$$L_{x^m y^n}(x, y, \sigma) = (\frac{\partial^{m+n}}{\partial x^m \partial y^n} g_\sigma * f)(x, y) \tag{3.14}$$

3.2.3 Espace d'échelle laplacien

L'opérateur laplacien, noté ∇^2, est défini comme la somme des dérivées secondes par rapport à chacune des variables :

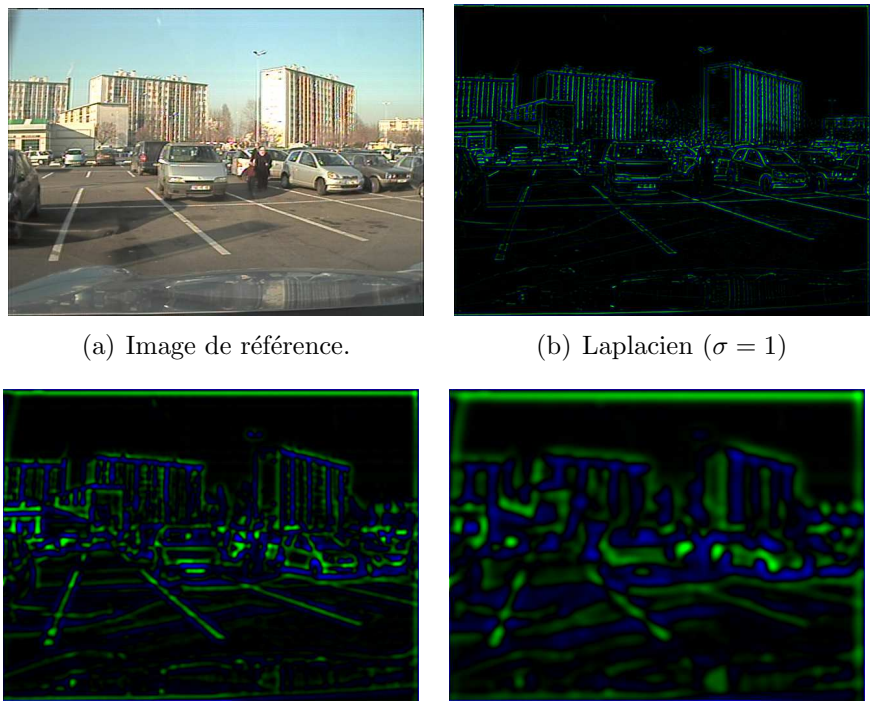
$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{3.15}$$

On peut donc définir l'espace d'échelle laplacien en utilisant cet opérateur.

Définition 3 *L'espace d'échelle laplacien, noté* $\nabla^2 L$*, associé à l'espace d'échelle* L *est défini par l'équation suivante :*

$$\nabla^2 L = L_{x^2} + L_{y^2} \tag{3.16}$$

Cet espace est particulièrement intéressant car il permet d'accentuer les zones de fort contrastes. En effet les zones uniformes ont des dérivées secondes nulles alors que les zones claires entourées de pixels foncés ou inversement auront une dérivée seconde de forte amplitude. De plus, contrairement aux dérivées premières, le laplacien est maximal aux centres des zones contrastées, ce qui permet d'accentuer la structure des objets plutôt que leurs frontières (voir Figure 3.3).



(a) Image de référence. (b) Laplacien ($\sigma = 1$)

(c) Laplacien ($\sigma = 4$) (d) Laplacien ($\sigma = 8$)

FIG. 3.3 – Image du laplacien pour différents niveaux d'échelles σ. Les valeurs nulles sont représentées en noir, les valeurs positives en bleu et les valeurs négatives en vert. On remarque que les structures des éléments ressortent bien, particulièrement lorsque la valeur de σ correspond à la taille des caractéristique des éléments.

3.2.4 Echelle caractéristique

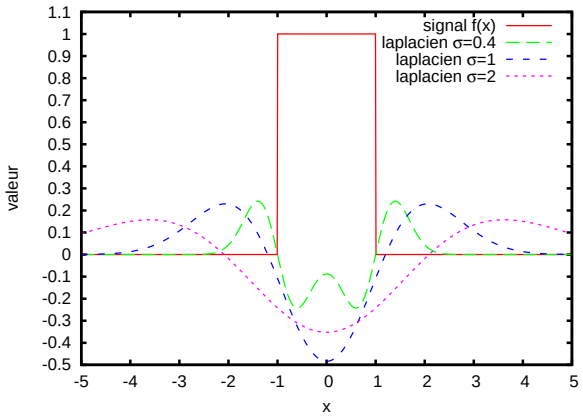
Comme nous l'avons vu dans les parties précédentes, l'espace d'échelle fournit des outils permettant de représenter une image et ses dérivées à différentes échelles, cependant, dans de nombreaux cas, on souhaite revenir dans un espace bidimensionnel, il est alors nécessaire de sélectionner automatiquement une échelle caractéristique pour chaque pixel de l'image.

Afin de sélectionner automatiquement cette échelle, Linderberg [Lin98] a proposé d'utiliser les maxima locaux des dérivées normalisées :

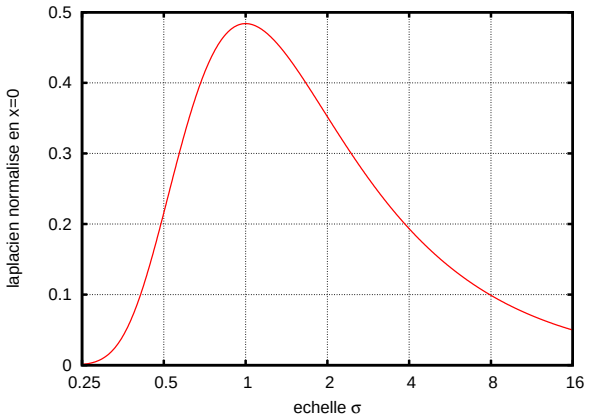
$$L'_{x^m y^n}(x, y, t) = \sigma^{m+n} L_{mn}(x, y, t) \tag{3.17}$$

Avec cette définition, les échelles pour lesquelles la dérivée normalisée est un extremum (dans l'axe des échelles) correspondent aux échelles caractéristiques des variations spatiales.

Si l'on considère en particulier le laplacien normalisé, que l'on notera $\nabla^2_{norm} L = \sigma^2 \cdot \nabla^2 L$, on constate que pour un objet uniforme contrasté par rapport à l'arrière plan, l'échelle caractéristique au centre de cet objet correspond à la dimension de l'objet dans l'image. La Figure 3.4 illustre bien ce phénomène, on peut y voir l'espace d'échelle laplacien normalisé correspondant à un signal de type "porte", on constate que la valeur maximale du laplacien normalisé est atteinte au centre de la porte et pour une échelle correspondant à la demi-largeur de la porte. De cette manière, on peut donc associer une taille caractéristique à chaque point d'une image en évaluant le laplacien à différentes échelles et en recherchant l'échelle qui maximise ce laplacien (voir Figure 3.5). Cette propriété est intéressante car on peut obtenir de cette manière une information de taille des objets présents dans une image sans avoir à segmenter l'image pour extraire les différents éléments.



(a) Laplacien normalisé à différentes échelles pour un signal 1D de type porte.



(b) Amplitude du laplacien en fonction de l'échelle en x=0.

FIG. 3.4 – Laplacien normalisé pour une signal 1D de type "porte", le laplacien est maximal au centre de la porte et pour une échelle correspondant à la demi-largeur de la porte. Cette échelle correspond à l'échelle caractéristique du signal.

(a) Image de test et échelle caractéristique prise en 3 points particuliers (centre du piéton, poteau de gauche et voiture à droite). Les positions des trois points et de l'échelle caractéristique correspondante sont représentées par trois cercles rouges.

(b) Image des échelles caractéristiques, une couleur claire correspond à une grande échelle.

FIG. 3.5 – Echelle caractéristique pour chaque pixel d'une image 3.5(b) et pour quelques points particuliers 3.5(a). On peut constater que l'échelle caractéristique correspond à la taille caractéristique des objets.

3.2.5 Construction de l'espace d'échelle

En pratique, il est impossible de construire un espace d'échelle continu, l'espace d'échelle calculé doit donc être discrétisé, à la fois spatialement et en échelle. De plus afin de garantir des performances temps réel, il sera nécessaire de minimiser la taille de l'espace d'échelle calculé.

Tout d'abord, dans le domaine discret, le produit de convolution en deux dimensions s'écrit sous forme de somme :

$$(h_1 * h_2)(x, y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} h_1(x-n, y-m) h_2(n, m)$$

Espace gaussien

On peut noter que, du fait de sa symétrie circulaire, la fonction gaussienne est séparable, c'est-à-dire qu'elle peut s'écrire en produit de deux composantes unidimensionnelles. Le produit de convolution avec la gaussienne peut alors se décomposer en deux convolutions unidimensionnelles avec une gaussienne 1D. L'espace d'échelle discretisé peut alors s'écrire de la manière suivante :

$$g_\sigma * f = G_t^y * (G_t^x * f)$$

où G_σ^x et G_σ^y représentent les noyaux gaussiens unidimensionnels à l'échelle σ dans les dimensions respectives x et y. Cela permet d'effectuer deux convolutions unidimensionnelles au lieu d'une convolution bidimensionnelle. Si n est la taille du noyau de convolution discrétisé, on effectue alors $2 * n$ opérations au lieu de n^2, ce qui permet une grande réduction du temps de calcul.

Plusieurs approximations du noyau gaussien peuvent être utilisées :

Noyau gaussien échantillonné Tout d'abord, l'approche la plus évidente est d'utiliser un noyau gaussien échantillonné :

$$G_\sigma^x(x) = \frac{1}{\sqrt{2\pi t}} e^{-\frac{x^2}{2\sigma^2}}$$

Le noyau est alors tronqué sur les bords, ce qui donne un filtre à réponse impulsionelle finie :

$$L(x, \sigma) = \sum_{n=-R}^{R} f(x-n) G_\sigma^x(n)$$

Le paramètre R doit être suffisamment grand pour que l'erreur due à la réduction du filtre soit faible. En pratique on choisit une valeur de R proportionelle à σ. Pour $\sigma = 1$, on choisit généralement R entre 3 et 6.

Noyau gaussien discret Lindeberg a redéfini dans [Lin90] la notion d'espace d'échelle dans le cas discret et a ainsi défini un noyau qui vérifie la définition discrète de cet espace. Le noyau s'écrit alors en utilisant la fonction de Bessel d'ordre entier :

$$T_\sigma(x) = e^{-\sigma^2} I_x(\sigma^2)$$

où I_x est la fonction de Bessel d'ordre entier.

Filtres récursifs Une implantation de filtres récursifs gaussien a été proposé par Vliet *et al.*[VTV98]. En dimension 1, ils consistent à filtrer le signal dans les deux directions successivement en utilisant une équation récursive, de gauche à droite :

$$v(x) = \alpha f(x) - \sum_{i=1}^{N} b_i * v(x-i)$$

puis de droite à gauche :

$$y(x) = \alpha v(x) - \sum_{i=1}^{N} b_i * y(x+i)$$

où b_i sont des coefficients réels et $\alpha = 1 + \sum_{i=1}^{N}$ et N est un entier, pour une bonne précision, on choisit en général $N = 5$.

L'intérêt de ces filtres est que le nombre d'opérations est indépendant de la variance du filtre, ils peuvent donc être utilisés pour calculer rapidement les grandes échelles.

Filtres binomiaux Les filtres binomiaux sont définis par auto-convolution du filtre $[1, 1]$. Ces filtres sont intéressant car ce sont les filtres constitués de valeurs entières qui approximent le mieux la fonction gaussienne (en terme de moindres carrés). Ces filtres n'utilisent que des calculs entiers et peuvent donc être utilisés dans des processeurs embarqués n'utilisant pas de nombres flottants.

La Figure 3.6 montre les différences entres les coefficients des différents filtres.

Afin de minimiser les approximations et pour conserver au mieux les propriétés de l'espace d'échelle dans le cas d'une image numérique (et donc

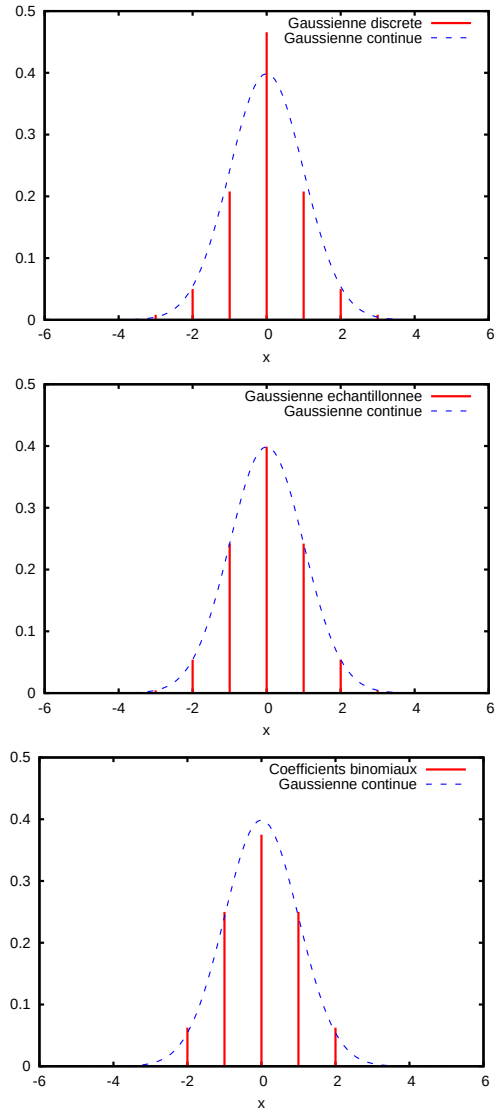


FIG. 3.6 – De haut en bas : coefficients du filtre gaussien discret, du filtre gaussien échantillonné et du filtre binomial (normalisé). Le filtre gaussien discret colle moins bien la gaussienne continue mais conserve mieux les propriétés de l'espace d'échelle dans le cas discret. Le filtre binomial est une bonne approximation de la gaussienne par des valeurs entières.

discrétisée), nous avons utilisé le filtre gaussien discret. Cependant, pour conserver une taille de filtre constante pour toutes les échelles et pouvoir effectuer les calculs en temps réel, nous avons choisi une approche pyramidale [CR03]. Une telle approche consiste à construire l'espace d'échelle en divisant

la taille au fur et à mesure qu'on augmente l'échelle. L'espace d'échelle est donc composé de plusieurs niveaux de résolutions différentes, un niveau pouvant comprendre plusieurs sous-niveaux d'échelle de même résolution (voir Figure 3.7). On obtient alors un espace d'échelle avec un espacement exponentiel de l'échelle.

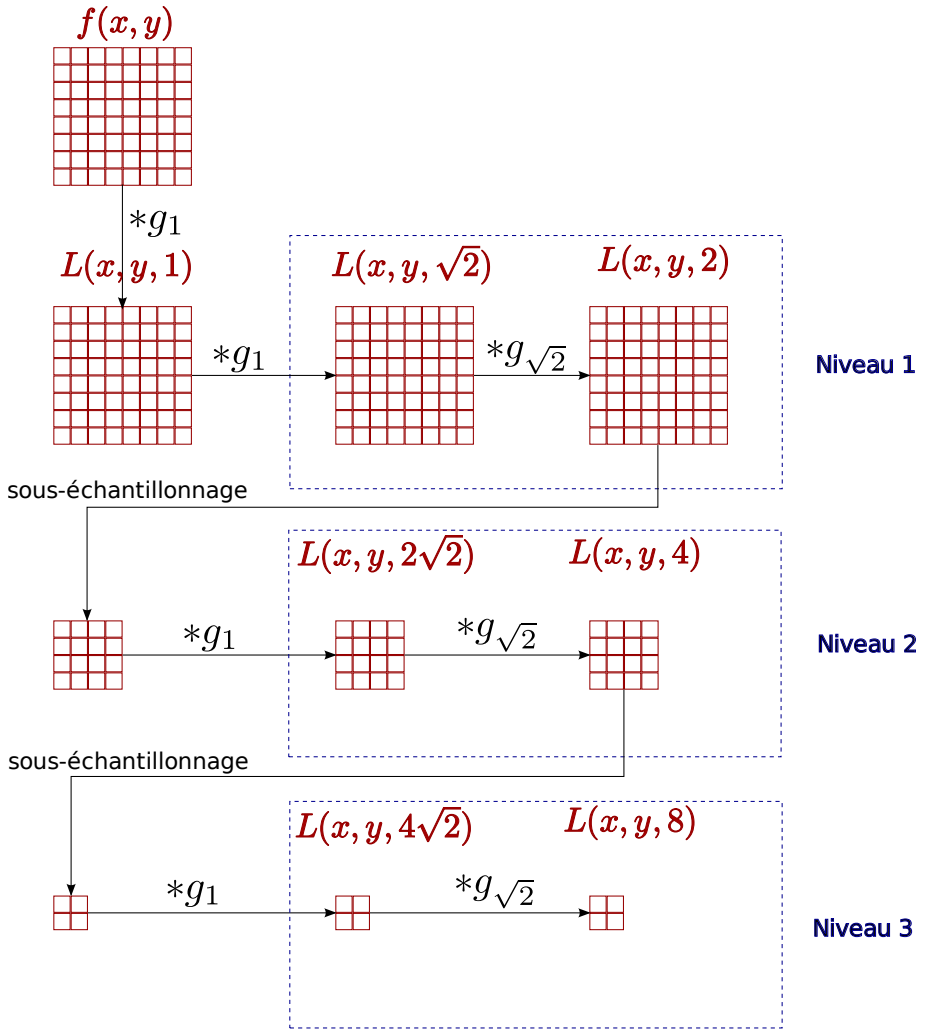


FIG. 3.7 – Calcul de l'espace d'échelle pyramidal. L'algorithme consiste à appliquer successivement des convolutions avec un noyaux gaussien d'échelle $\sigma = 1$ et $\sigma = \sqrt{2}$ puis à sous-échantillonner le signal obtenu pour passer au niveau suivant. On obtient ainsi un espace d'échelle avec un espacement exponentiel des échelles. Du fait du sous-échantillonnage à chaque niveau, le coût de calcul est en $\mathcal{O}(n)$ où n est le nombre de pixel de l'image originale.

Dérivées de l'espace d'échelle

Etant donné que l'on utilise une approximation discrète de l'espace d'échelle, nous pouvons utiliser les opérateurs différentiels discrets [Lin93] :

$$L_x(x,y,\sigma) = \frac{1}{2}(L(x+1,y,\sigma) - L(x-1,y,\sigma)) \quad (3.18)$$

$$L_{x^2}(x,y,\sigma) = L(x+1,y,\sigma) - 2L(x,y,\sigma) + L(x-1,y,\sigma) \quad (3.19)$$

Les dérivées premières et secondes de l'espace d'échelle s'obtiennent donc en convoluant chaque signal par les noyaux suivants :
$N_x = [-1, 0, 1]$ et $N_{x^2} = [1, -2, 1]$
En deux dimensions, nous pouvons utiliser les propriétés de séparabilité pour obtenir toutes les dérivées en x et y :

$$L_{x^i y^j} = L_{x^i}(L_{y^j}(x,y,\sigma)) \quad (3.20)$$

L'espace d'échelle laplacien se calcule de la même manière :

$$Lap(x,y,\sigma) = L_{x^2}(x,y,\sigma) + L_{y^2}(x,y,\sigma) \quad (3.21)$$

3.3 Détection des pics

Une première approche pour détecter des éléments visuels de manière stable par rapport à différents types de transformations est de se tourner vers les points d'intérêts. De nombreux types de détecteurs existent dans la littérature pour extraire de tels points. Tout d'abord, les points d'intérêt de Harris [HS88] ont été conçus pour détecter les positions où le signal représenté par l'image varie le plus. Ce détecteur utilisant les dérivées premières localise principalement les coins ou les extrémités des objets. Ce détecteur est très robuste aux rotations, aux changements d'illuminations mais peu robuste aux variations d'échelle. De plus ces points sont localisés sur les frontières des objets physiques, ce qui n'est pas intéressant pour localiser précisément des obstacles.

Afin d'améliorer la robustesse au changement d'échelle, Mikolajczyk et Schmid [MS01] ont amélioré le détecteur de Harris en utilisant l'espace d'échelle et le laplacien. Ce détecteur, appelé Harris/Laplace, consiste à chercher itérativement les maxima spatiaux de la mesure de Harris pour une échelle donnée, puis à optimiser l'échelle en cherchant le maximum local du laplacien en échelle. Bien que ce détecteur soit bien plus robuste que le détecteur de Harris pour les changements d'échelle, il localise toujours les

points correspondant à des coins et ne représente donc pas correctement la structure des objets.

Un autre type de détecteur permet de détecter les *blobs*, ces types de points que l'on peut traduire par *tâches* ou *gouttes* en francais correspondent à des zones claires ou foncées qui ressortent nettement, comme par exemple une tâche noire sur fond blanc ou inversement. Pour détecter de tels points, la première solution consiste à utiliser le laplacien de gaussienne dans l'espace d'échelle (voir Equation 3.16) qui donne une réponse positive sur les taches noires et négative sur les taches claires. Pour être invariant à l'échelle, il est nécessaire d'utiliser les propriétés énoncées en 3.2.4 et de normaliser le laplacien par l'échelle :

$$\nabla^2_{norm} L = \sigma^2 (L_{x^2} + L_{y^2}) \tag{3.22}$$

La différence de gaussienne (DOG) est aussi souvent utilisée [Low99] car elle approxime bien le laplacien tout en étant plus rapide à calculer. Cette approximation est fondée sur l'équation de diffusion ($\nabla^2 L = \frac{\partial L}{\partial \sigma}$).

Un exemple de détection est illustré par la Figure 3.8. On peut voir que les objets contrastés et isotropes sont bien détectés au centre, mais les objets allongés sont très mal détectés.

3.4 Détection des crêtes et segments de crêtes

Le problème avec les points d'intérêts considérés précédemment est le fait qu'ils sont adaptés aux formes isotropes. Etant donné que les points recherchés correspondent à des maxima locaux d'une certaine fonction dans toutes les directions[2] (x, y, et σ), les objets qui génèrent une forte réponse sur toute une ligne ne vont pas être détectés, ou vont donner des résultats instables. Comme on peut le voir sur la Figure 3.9, pour un segment d'une certaine épaisseur, la détection des extrema locaux du laplacien fait ressortir les deux extrémités (à l'échelle correspondant à l'épaisseur du segment) et le centre du segments (à l'échelle correspondant à la longueur du segment). Cependant, aucun des points ne décrit de manière globale le segment.

Plusieurs solutions permettent de faire ressortir la structure et la forme des objets, la première solution est de détecter les contours en utilisant par exemple le détecteur de Canny [Can86], cependant les contours décrivent uniquement les frontières entre les objets. Pour reconstituer l'objet il est donc

[2]Le paramètre σ correspondant au paramètre d'échelle rajoute une troisième dimension à l'image.

FIG. 3.8 – Résultat de la détection de points d'intérêts (extrema du laplacien dans l'espace d'échelle). Les objets contrastés (pieton, vitres des voitures, etc.) sont bien détectés mais les points détectés caractérisent mal la structure des objets. De plus les objets allongés (lignes du sol, plaques des voitures, etc.) sont mal détectés.

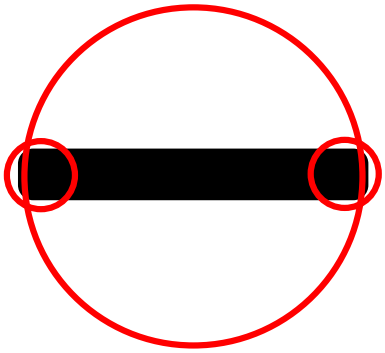


FIG. 3.9 – Illustration de la détection de points d'intérêts (extrema du laplacien) sur un segment. Un point d'intérêt est détecté à chaque extrémité et au millieu du segment.

nécessaire de rechercher les contours fermés, ce qui peut s'avérer difficile et très coûteux. Une seconde solution consiste à détecter la courbe médiane qui représente le mieux cet objet, cela est possible par la détection des crêtes,

que nous allons définir dans la section suivante.

3.4.1 Définition des crêtes

Dans la vie courante, une crête est un chemin pour lequel les deux côtés adjacents à cette ligne "redescendent". Dans une image, une ligne de crête peut être définie de manière identique si l'on considère l'intensité lumineuse comme une hauteur. C'est une courbe dont l'ensemble des points est localement maximal dans au moins une direction. De manière identique, une vallée est définie comme un ensemble de points qui sont des minima locaux. Avec cette définition intuitive, une ligne de crête ou une vallée peut alors être considérée comme une approximation de l'axe médian d'un objet allongé.

On peut définir formellement un point de crête en utilisant les opérateurs différentiels de l'espace d'échelle :

Soit $X = (x, y, \sigma)$ un point dans l'espace d'échelle.

On note $\mathcal{H}$ la matrice Hessienne de l'image dans l'espace d'échelle L en X :

$$\mathcal{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial xy} \\ \frac{\partial^2 f}{\partial xy} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

On note λ_1 et λ_2 les deux valeurs propres de cette matrice, tel que $|\lambda_1| >= |\lambda_2|$ et $\vec{u}_1$, $\vec{u}_2$ leurs vecteurs propres associés. On peut noter que $\vec{u}_1$ représente la direction de la plus forte courbure.

On note L_1 la dérivée de l'image dans la direction de $\vec{u}_1$

Définition 4 *X est un point de crête si et seulement si :*

$$L_1(X) = 0 \text{ et } \lambda_1 \leq 0 \tag{3.23}$$

Définition 5 *X est un point de vallée, si et seulement si :*

$$L_1(X) = 0 \text{ et } \lambda_1 \geq 0 \tag{3.24}$$

3.4.2 Détection des points de crêtes

Dans une image numérique, le critère de la dérivée nulle n'est pas directement exploitable. Pour un point X, il est plus judicieux de parcourir les pixels voisins dans la direction de la plus forte courbure pour vérifier que le pixel est bien un extrema local. Cependant, au lieu d'utiliser directement l'espace d'image, Tran et Lux [TL04] ont proposé d'utiliser le laplacien normalisé de l'image (cf. Equation 3.22) qui donne une forte réponse sur

les points de crêtes. Cependant, le filtrage par le laplacien produit aussi des artefacts, appelés “fausses crêtes“ de chaque côté de la ligne de crête. Ces fausses crêtes peuvent être facilement enlevées soit en calculant le gradient de l'image (qui est élevé pour les fausses crêtes), soit en regardant si le laplacien change de signe des deux côtés de la crête (dans le cas positif, c'est bien un point de crête). Ensuite, après avoir détecté chaque point de crête séparément, il est nécessaire d'associer les points qui appartiennent à la même ligne de crête pour pouvoir calculer différents paramètres comme la longueur de la crête, le centre de gravité, etc.

La méthode classique pour détecter les lignes de crêtes peut alors être décrite par l'algorithme suivant.

Algorithme 1 Détection des lignes de crêtes

1. Calcul de l'espace d'échelle gaussien et du laplacien normalisé.
2. Pour chaque pixel, si le laplacien est supérieur à un certain seuil :
   (a) Calcul de la matrice Hessienne, des valeurs propres et des vecteurs propres associées.
   (b) Si le point correspond à un extremum local du laplacien dans la direction du vecteur propre principal (associé à la plus grande valeur propre), alors c'est un point de crête.
3. Linkage des points par recherche de connexité.

Cependant, la phase de linkage des points peut être problématique : si l'image est perturbée par du bruit, on peut avoir des erreurs de fusion ou de séparation d'objets. Pour résoudre ce problème nous allons mettre au point un détecteur plus local mais plus spécialisé pour un certain type d'objets.

3.4.3 Détection des segments de crête

Un segment de crête est un ensemble contigu de points de crête alignés. Dans cette partie nous allons voir comment paramétriser de tels segments puis nous définirons l'algorithme et les mesures utilisés pour la détection.

Paramétrisation des segments de crête

Un segment de crête S peut être représenté de la manière suivante :
- Position du milieu du segment dans l'espace d'échelle $\vec{c} = (x_c, y_c, \sigma_c)$
- Direction et longueur du demi-segment, représenté par le vecteur entre le centre et une extrémité : $\vec{r} = (x_r, y_r, \sigma_r)$

Ces paramètres sont illustrés par la Figure 3.10. Dans la suite on notera $S = \begin{pmatrix} \vec{c} \\ \vec{r} \end{pmatrix}$, et $\vec{u} = \frac{\vec{r}}{\|\vec{r}\|}$ représentera le vecteur directeur du segment.

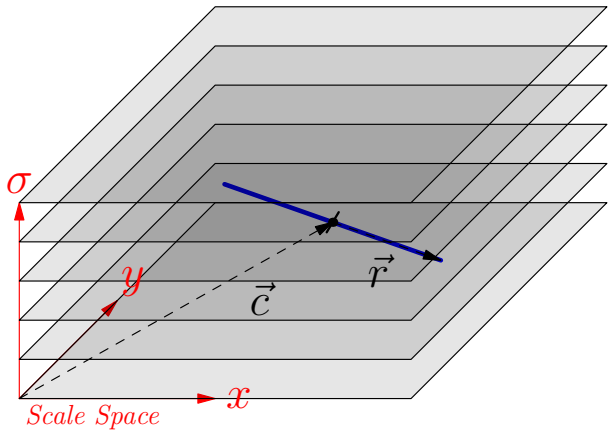


FIG. 3.10 – Paramétrisation d'un segment de crête : un segment est représenté par son centre $\vec{c}$ et son demi-segment $\vec{r}$ (orientation et taille).

Fonction de score

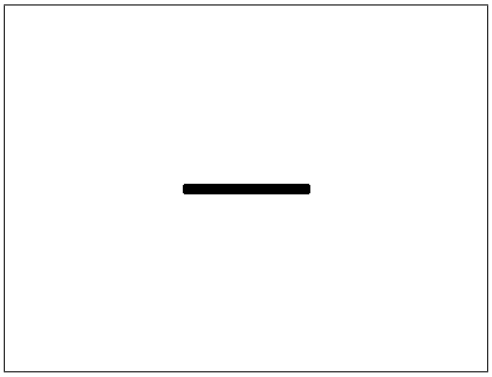
Pour détecter les segments, nous allons chercher à maximiser une fonction dans l'espace des paramètres des segments. Nous rappelons que les points de crêtes sont des maxima locaux du laplacien normalisé $\nabla^2_{norm} L$ dans une direction. La fonction à maximiser va donc correspondre à la somme du laplacien le long du segment :

$$f_1(S) = \int_{l=-||\vec{r}||}^{||\vec{r}||} |\nabla^2_{norm} L(\vec{c} + l \cdot \vec{u})| dl \tag{3.25}$$

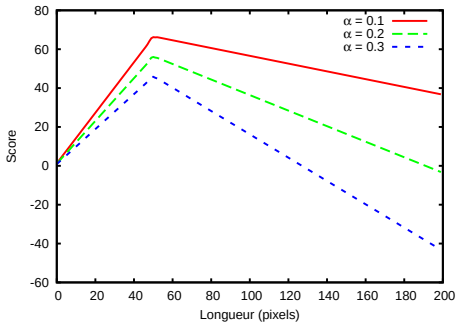
Cependant, si l'on utilise ce score tel quel, pour un centre et une orientation fixés, la fonction de score est croissante en fonction de la longueur du segment, il faut ajouter un facteur de correction pour assurer que le score diminue avec la longueur du segment :

$$f_2(S) = f_1(S) - \alpha||\vec{r}|| \tag{3.26}$$

où α est un paramètre à déterminer, il représente la valeur minimale du laplacien pour lequel nous souhaitons détecter un segment de crête. Un exemple de score en fonction de la longueur du segment et du paramètre α est illustré dans la Figure 3.11. On peut voir que plus α est élevé, plus la fonction est piquée, mais plus le score est faible.



(a) Image utilisée pour calculer la fonction de score.



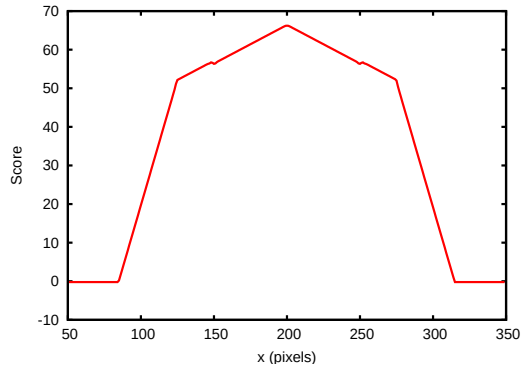
(b) Résultat du score f_2 en fonction de la longueur du segment évalué.

FIG. 3.11 – Score f_2 (voir Equation 3.26) associé au segment de crête centré et aligné avec la ligne noire en fonction de la longueur du segment et pour différentes valeurs du paramètre α

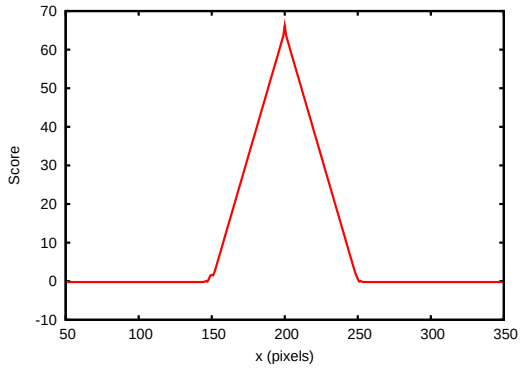
Ensuite, pour améliorer la localité spatiale, il faut ajouter une correction supplémentaire pour faire diminuer le score si le segment n'est pas bien centré avec le segment de crête physique. Pour cela, nous allons ajouter une contrainte de symétrie. Nous allons imposer que l'intensité du laplacien soit symétrique par rapport au centre du segment. La fonction de score devient :

$$f_3(S) = f_2(S) - \int_{l=-||\vec{r}||}^{||\vec{r}||} |\nabla^2_{norm} L(\vec{c} + l \cdot \vec{u}) - \nabla^2_{norm} L(\vec{c} - l \cdot \vec{u})| dl \quad (3.27)$$

L'effet de la contrainte sur la fonction de score est illustré par la Figure 3.12, on remarque que la fonction de score est beaucoup plus piquée sur le centre du segment physique avec la contrainte de symétrie.



(a) Score f_2 (sans contrainte de symétrie)



(b) Score f_3 (avec contrainte de symétrie)

FIG. 3.12 – Score maximal du segment en fonction de la position du centre et avec une orientation fixée avec et sans contrainte de symétrie. On constate que le pic est beaucoup plus marqué avec la contrainte de symétrie, ce qui permet d'avoir une meilleure robustesse.

Réduction de la dimensionalité

Pour détecter les segments de crête dans une image, il faut rechercher les maxima locaux de cette fonction dans l'espace des paramètres. Cependant, compte tenu de la dimensionalité (6 dimensions), une recherche exhaustive sera extrêmement coûteuse. Nous allons modifier l'algorithme pour réduire le coût de la recherche.

Tout d'abord, nous allons limiter la recherche aux segments de crête appartenant à des plans d'échelle fixes dans l'espace d'échelle (c'est à dire $\vec{r} = (x_r, y_r, 0)$). Cela revient à se limiter aux motifs dont l'épaisseur ne varie pas le long de leur axe principal. Ensuite, il est possible d'évaluer la direction du segment de crête en utilisant la matrice Hessienne. Le vecteur propre

associé à la plus grande valeur propre de cette matrice est dirigé dans le sens de la plus forte courbure. La direction de la ligne de crête est orthogonale à ce vecteur. Toutefois, du fait de la discrétisation de l'espace d'échelle, les dérivées secondes peuvent êtres légèrement différentes de la direction du segment de crête. Il faut donc considérer également un voisinage autour de la direction calculée pour obtenir de meilleurs résultats.

L'algorithme 2 résume la méthode de détection des segments de crêtes. La sortie de cet algorithme est illustrée à la Figure 3.13.

Algorithme 2 Détection des segments de crête.

1. Calcul de l'espace d'échelle gaussien ;
2. Calcul des dérivées premières et secondes ;
3. Calcul de l'espace laplacien normalisé et élimination des "fausses" crêtes ($\frac{Laplacien}{Gradient} < seuil$) ;
4. Pour chaque pixel $\vec{p} = (x, y, \sigma)$ de l'espace d'échelle dont le laplacien est supérieur au seuil :
   (a) Evaluation de la direction de la crête $\vec{u}$ à l'aide de la matrice Hessienne ;
   (b) Evaluation des scores $f_3 \begin{pmatrix} \vec{p} \\ l \cdot \vec{u} \end{pmatrix}$ pour $l \in]0, l_{max}]$
   (c) Stockage du meilleur score et des paramètres correspondants.
5. Recherche des maxima locaux des scores stockés à l'étape 4c.

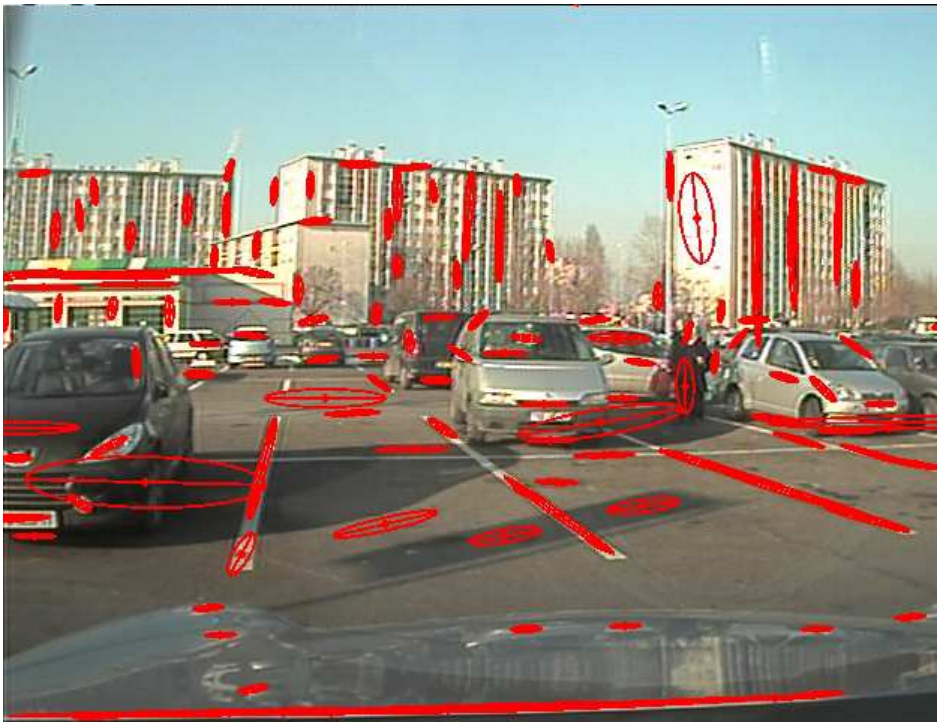


FIG. 3.13 – Exemple de détection des segments de crête. Les segments rouges représentent les paramètres spatiaux des segments de crêtes détectés, les ellipses permettent de visualiser l'échelle des segments (longueur de l'axe orthogonal au segment). On peut voir que les lignes du sol sont très bien détectées à l'échelle correspondant à leur largeur ; à plus grande échelle, on détecte bien les objets plus larges, tel que les bâtiments, le piéton et les ombres.

3.5 Conclusion

Ce chapitre a présenté un détecteur de segment de crête qui permet d'extraire principalement les objets allongés contrastés par rapport à leur arrière plan. Ce détecteur utilise la notion d'espace d'échelle, ce qui le rend invariant au changement d'échelle (changement de taille des objets). De plus, étant donné que l'algorithme consiste à maximiser un score dans l'espace d'échelle, la taille caractéristique des objets est automatiquement détectée. Les performances de ce détecteur seront analysées plus en détail dans le Chapitre 6.2.1. Nous verrons que ce détecteur est bien robuste aux différentes transformations de l'image (rotation, changement d'échelle, *etc...*) et bien adapté pour la détection des éléments urbains. Toutefois ce détecteur peut également détecter des éléments tels que des marques sur le sol mais aussi

des espaces vides correspondant à des zones contrastées entre deux éléments physiques (espace libre entre deux piétons par exemple). Il serait donc utile de filtrer ces éléments pour diminuer le nombre de faux positifs.

Dans le chapitre suivant, nous allons voir comment suivre ces objets au cours du temps afin d'avoir une estimation de leur trajectoire et leur vitesse dans l'espace d'échelle, ce qui est une information précieuse pour prédire les collisions ou le danger.

Chapitre 4

Suivi des structures

Une fois la détection des structures effectuée, il est nécessaire de suivre ces objets d'une image à l'autre afin de pouvoir mesurer leur mouvement et effectuer tout type de mesures différentielles. Pour effectuer une tel suivi en temps réel, nous allons utiliser un filtre bayésien que nous allons définir.

4.1 Le filtragre bayésien

Avant de définir les filtres bayésiens, il est nécessaire de définir la notion de processus markovien :

Définition 6 (Processus de Markov) *Un processus dépendant du temps et caractérisé par un état X est dit markovien d'ordre 1 si son état courant X_t ne dépend que de son état précédant X_{t-1}. Un processus de Markov est dit caché si son état n'est pas directement observable.*

Définition 7 (Filtre bayésien) *Un filtre bayésien permet d'estimer l'état X_t d'un processus de Markov au cours du temps en fonction des observations réalisées $Z_O..Z_t$.*

Avec cette définition, l'état du système peut s'écrire sous forme récursive :

$$P(X_t|Z_0...Z_t) \propto P(Z_t|X_t) \int_{X_{t-1}} P(X_t|X_{t-1})P(X_t|Z_{t-1}...Z_0) \tag{4.1}$$

On peut noter que la définition du filtre bayésien dépend complètement de la définition de deux modèles :

Le modèle de transition : $P(X_i|X_{i-1})$ qui permet de prévoir l'état du système en fonction de l'état précédant. Il s'agit en général d'un modèle dynamique prenant en compte une incertitude introduite par des variables cachées.

Le modèle d'observation : $P(Z_i|X_i)$ qui permet de décrire l'observation Z_i en fonction de l'état du système.

Parmi les filtres bayésiens, deux spécialisations sont particulièrement employées : le filtre de Kalman et le filtre à particules.

4.1.1 Le filtre de Kalman

Le filtre de Kalman [Kal60, Cro89, WB95] est un filtre bayésien dont toutes les formes paramétriques sont gaussiennes et dont le modèle de transition et d'observation sont linéaires. Avec ce filtre, on représente l'état du système par une moyenne $\bar{X}_t$ et une matrice de covariance P_t. Pour décrire l'évolution du système, on pose :

$$X_{t+1} = A \cdot X_t + V$$

où A est la matrice de transition qui relie l'état précédant à l'état actuel et V est une variable aléatoire gaussienne centrée en 0 et de covariance Q. Cette variable traduit l'incertitude associée au modèle dynamique que l'on apelle généralement bruit de process.

On considère que l'observation Z_t est liée à l'état du système par la relation :

$$Z_t = B \cdot X_t + W$$

où B est la matrice d'observation qui relie l'état X_t à la mesure Z_t et W est une variable aléatoire gaussienne centrée en 0 et de covariance R. Cette variable représente l'incertitude provenant des capteurs, appelée aussi bruit de mesure.

L'état du système peut alors être exprimé à tout moment par les formules suivantes :

$$\begin{aligned}
X_{pred} &= A \cdot \bar{X}_t && (4.2)\\
P_{pred} &= A \cdot P_t \cdot A^T + Q && (4.3)\\
K &= P_{pred} \cdot B^T \cdot (B \cdot P_{pred} \cdot B^T + R)^{-1} && (4.4)\\
\bar{X}_{t+1} &= X_{pred} + K \cdot (Z_{t+1} - B \cdot X_{pred}) && (4.5)\\
P_{t+1} &= (I - K \cdot B) \cdot P_{pred} && (4.6)
\end{aligned}$$

où X_{pred} et P_{pred} représentent l'état prédit à partir du modèle de transition, K est appelé gain de Kalman, et $\bar{X}_{t+1}$ et P_{t+1} représentent l'état estimé après l'observation Z_{t+1}.

Une des principales limites du filtre de Kalman est le fait que les deux modèles de transition et d'observation doivent être linéaires. Pour continuer d'utiliser ce filtre lorsque ces modèles ne sont pas linéaires, deux extensions du filtre de Kalman sont donc apparus :

Le filtre de Kalman étendu (EKF) consiste à linéariser ces modèles au voisinage de l'état estimé. Toutefois, la convergence d'un tel filtre n'est pas assurée, il s'agit uniquement d'une convergence locale, lorsque les modèles sont fortement non linéaires, cette approximation mènera donc à de mauvais résultats.

Le filtre de Kalman "unscented" : dans le cas de fonction fortement non linéaire, Julier et Uhlmann [JU97] ont développé une extension du filtre de Kalman qui utilise un jeu de points choisis de manière optimale, qui sont propagés en utilisant la fonction de transition. Le modèle d'observation est alors appliqué sur cet ensemble de points pour estimer la moyenne et la covariance de la variable d'état.

4.1.2 Le filtre particulaire

Le filtre particulaire [AMGC02], également connu par l'algorithme de condensation [IB98], consiste à représenter la distribution de probabilité $P(X_t|Z_{t-1}...Z_0)$ par un ensemble de particules avec un poids associé. Chaque particule peut être vue comme une hypothèse de l'état estimé, et le poids correspond à la probabilité de cette hypothèse.

De manière formelle, si on note $\{p_t^i\}_{i=1}^n$ l'ensemble des n particules à l'instant t et $\{w_t^i\}_{i=1}^n$ leur poids associé (les poids sont normalisés, de telle sorte que $\sum_{i=1}^n w_t^i = 1$), alors on peut approximer la densité de probabilité $P(X_t|Z_{t-1}...Z_0)$ par :

$$P(X_t|Z_{t-1}...Z_0) \approx \sum_{i=1}^{n} w_t^i \delta(X_t - p_t^i) \tag{4.7}$$

Où $\delta(x) = \begin{cases} 1 \ si \ x = 0 \\ 0 \ sinon \end{cases}$

Il faut noter que la distribution obtenue n'est qu'une approximation de la vraie distribution, cependant aucune forme n'est imposée sur cette distribution (éventuellement multimodale), contrairement au filtre de Kalman qui impose une distribution gaussienne. Le nombre de particules

n joue un rôle important, plus ce nombre est élevé, meilleure sera l'approximation, mais plus le coût de calcul sera grand.

L'inférence du filtre à particule se divise en trois étapes :

1. Prédiction : On applique le modèle de transition (qui peut être non-linéaire) à chaque particule ;
2. Observation : On calcule le poids de chaque particule w_t en fonction de l'observation Z_t ;
$$w_t^i = w_{t-1}^i \cdot P(Z_t | X_t = p_t^i)$$
3. Ré-échantillonage : On supprime ou duplique certaines particules en fonction de la nouvelle distribution de probabilité.

Le ré-échantillonage permet d'éviter le problème de "dégénérescence" (lorsque, après plusieurs itérations, toutes les particules sauf une ont un poids négligeable). Cependant, il ne doit pas avoir lieu systématiquement pour éviter que toutes les particules ne s'agglutinent au même endroit (phénomène d'appauvrissement). Un compromis est possible avec une méthode d'échantillonage adaptatif. Cette méthode consiste à calculer un indicateur de dégénérescence, noté n_{eff} à l'aide de la formule suivante :

$$n_{eff} = \frac{1}{\sum_{i=1}^{n} (w_t^i)^2} \tag{4.8}$$

Cette valeur représente le nombre de particules qui ont un poids significatif. Le ré-échantillonage est effectué lorsque $n_{eff} < n_t$ où n_t est un seuil à définir à priori, en général on prend $n_t = \frac{n}{2}$

Plusieurs méthodes sont possibles pour le ré-échantillonage. Dans [AMGC02], les auteurs conseillent d'utiliser la méthode de ré-échantillonage systématique, du fait de son faible coût (voir Algorithme 3).

Toutefois, cette méthode n'est pas naturellement parallélisable. Avec les ordinateurs modernes possédant plusieurs processeurs, ou des processeurs graphiques permettant de paralléliser des tâches, il peut être plus judicieux d'utiliser une méthode plus facilement parallélisable afin d'accélérer les traitements, c'est pourquoi nous préférerons la méthode par rejet (Algorithme 4) qui consiste à tirer aléatoirement chaque nouvelle particule à partir de la distribution précédente. Avec cet algorithme, la même procédure peut être appliquée indépendament sur chaque particule et donc nous pouvons exploiter tous les processeurs disponibles en exécutant cette procédure en parallèle sur tous les processeurs.

La Figure 4.1 illustre les différentes étapes du filtre particulaire. Dans l'étape de prédiction, chaque particule est décalée en fonction du modèle de transition mais son poids n'est pas modifié. Dans la phase d'observation, seul

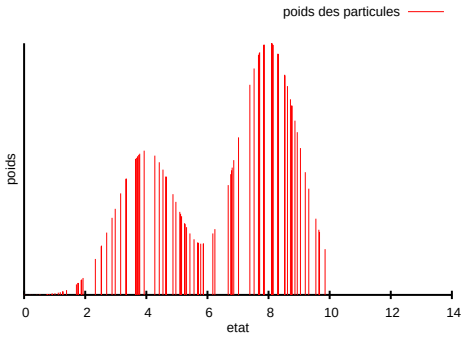
Algorithme 3 Ré-échantillonage systématique

1. Initialiser la fonction de répartition : $c_1 = 0$
2. **pour** $i = 2..n$ **faire**
   - $c_i = c_{i-1} + w_t^i$
3. **fin pour**
4. $i = 1$
5. Tirer le point de départ u_1 uniformément sur $[0, \frac{1}{n}]$
6. **pour**$j = 1..n$
   - $u_j = u_1 + \frac{j-1}{n}$
   - **tantque** $u_j > c_i$
     $i = i + 1$
   - **fin tantque**
   - $p_t^{j*} = p_t^i$
   - $w_t^j = \frac{1}{n}$
7. **fin pour**
8. $\{p_t^i\}_{i=1}^n = \{p_t^{i*}\}_{i=1}^n$

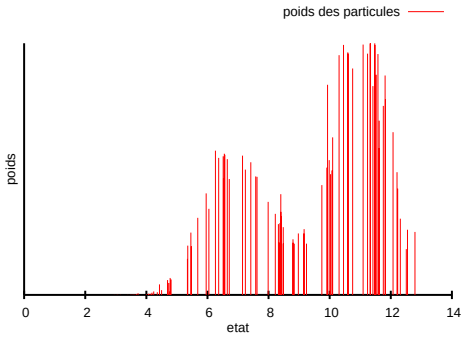
Algorithme 4 Ré-échantillonage par rejet

1. Calculer le poid maximum des particules : $w_{max} = \max_{i=1..n}\{w_t^i\}$
2. **pour** $i = 1..n$
   - **répéter**
     - Tirer j uniformément sur $[\![1, n]\!]$
     - $p_t^{i*} = p_t^j$
     - Tirer u uniformément sur $[0, w_{max}]$
   - **jusqu'à** $u < w_t^j$
   - $w_t^{i*} = \frac{1}{n}$
3. **fin pour**
4. $\{p_t^i\}_{i=1}^n = \{p_t^{i*}\}_{i=1}^n$
5. $\{w_t^i\}_{i=1}^n = \{w_t^{i*}\}$

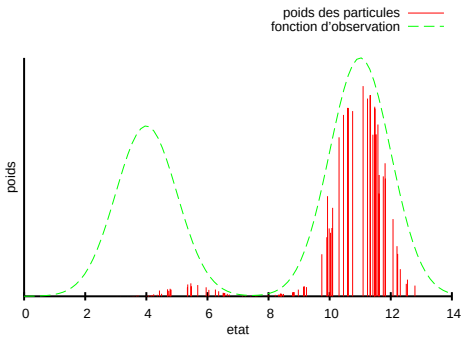
le poids des particules est modifié. Après l'étape de ré-échantillonage, toute les particules ont le même poids mais un certain nombre de particules ayant un poids faible avant cette étape ont disparu et les particules ayant un poids élevé ont été dupliquées.



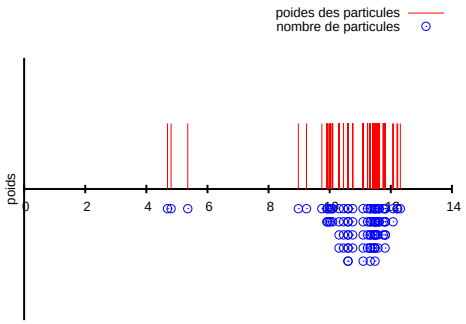
(a) Etat du filtre à particules à l'instant t



(b) Etat du filtre après prédiction, le modèle de transition a été appliqué à chaque particule, ici il s'agit d'une translation avec un bruit gaussien.



(c) Etat du filtre après observation à $t+1$. Le modèle d'observation est affiché avec la courbe en pointillé. Le poids des particules est modifié et résultent d'une fusion entre le poid précédant et la fonction d'observation.



(d) Etat du filtre après ré-échantillonage. Toutes les particules ont le même poids mais la répartition des particules est modifiée (cercles bleus) : les particules ayant précédement un poid très faible sont supprimées, alors que les particules ayant un poids élevé sont dupliquées.

FIG. 4.1 – Illustration des différentes étapes du filtre particulaire. L'état du filtre est représenté par un ensemble de particules avec un poids associé (batons rouges).

4.2 Suivi des segments de crête

4.2.1 Choix de la méthode

Le choix de la méthode de suivi dépend fortement de la nature du modèle d'observation et du modèle dynamique des objets suivis. Dans le filtre de Kalman, l'incertitude de l'observation et de la dynamique du système sont modélisées par une moyenne et une covariance. Ainsi, si les variables considérées sont bien de forme gaussienne, le filtre de Kalman correspond au choix optimal. Cependant, lorsque les fonctions d'observation et de transition sont plus compliquées (éventuellement multimodales), il est préférable de s'orienter vers un filtre à particules qui permet d'approximer tout type de distribution.

Dans notre cas, il est possible d'utiliser un filtre de Kalman en considérant les segments de crêtes détectés comme des observations. Cependant, de cette manière le problème de mise en correspondance est crucial. En cas de non détection ou de mauvaise association, il y a de forte chance que l'on diverge rapidement. Pour éviter des erreurs d'association, il faudrait mettre au point des descripteurs pour chaque segment de crête et les utiliser lors de la mise en correspondance. Il est également possible d'utiliser une filtre à association de données probabilistes (PDAF[1]) [BST75] pour tenir compte des multiples associations. Dans ce cas, le filtre tient compte de la présence de fausses alarmes et l'innovation du filtre de Kalman est remplacée par une somme pondérée des innovations associées à chaque hypothèse d'observation. Toutefois, la distribution estimée est toujours représentée par une distribution gaussienne, donc unimodale alors qu'il s'agit en fait d'une distribution multimodale.

En fin de compte en utilisant le filtre de Kalman ou ses extensions multi-hypothèses, les phases de détection et de mise en correspondance deviennent les phases cruciales du problème. Pour cela, le filtre à particules peut s'avérer être une bonne solution. En ayant la possibilité d'utiliser une fonction d'observation multimodale et non linéaire, on peut en effet utiliser directement une fonction similaire à la fonction de détection des segments de crête. Dans le cas où plusieurs hypothèses sont possibles, cela se traduit par la présence de plusieurs maxima locaux dans la fonction d'observation, mais cela ne pose pas de problèmes car les mauvaises hypothèses non stables devraient disparaître naturellement dans le temps. De plus, avec un filtre à particules, il est relativement aisé de modifier la fonction d'observation pour fusionner plusieurs modalités et prendre ainsi en compte différentes données

[1] *Probabilistic Data Association Filter*

comme la couleur, la valeur du laplacien, etc. Pour ces différentes raisons, nous avons choisi d'utiliser un filtre à particules pour effectuer le suivi des cibles dans l'image.

4.2.2 Représentation des cibles

Pour représenter l'état des cibles suivies, nous allons utiliser la même représentation que pour la détection (voir Partie 3.4.3), c'est-à-dire la position du centre du segment $\vec{c}$ et le demi-segment $\vec{r}$ représentant l'orientation et la longueur du segment. Comme dans le chapitre précédant, nous ne considérons pas la variation en échelle du segment, ce qui restreint la dimension des demi-segments à 2. De plus, nous allons ajouter la vitesse des cibles pour améliorer le suivi lorsque la caméra et/ou les objets bougent. Nous tiendrons compte uniquement de la vitesse du centre des segments, notée $\vec{v}$. Il est aussi possible de rajouter la vitesse du demi-segment, mais dans des applications où la caméra est embarquée dans un véhicule de type voiture, ce paramètre varie généralement peu et inclure cette variable dans l'état du filtre augmenterait la complexité sans y ajouter beaucoup d'informations.

Pour des raisons de linéarité, nous avons effectué le changement de variable $\rho = \frac{1}{\sigma}$ où σ est l'échelle du centre du segment. En effet, comme on le verra au chapitre suivant, lorsque l'on suit un objet 3D en se déplaçant à vitesse constante, l'inverse de l'échelle varie de manière linéaire. Le modèle de transition est donc plus simple avec ce changement de variable. Dans la suite, nous noterons $\vec{c} = (c_x, c_y, \sigma)^T$ et $\vec{v} = (v_x, v_y, v_\sigma)^T$ la position et la vitesse du centre dans l'espace d'échelle, et $\vec{c'}$ et $\vec{v'}$ les positions avec le changement de variable $\rho = \frac{1}{\sigma}$. L'état d'une cible est donc représenté par trois vecteurs :

$$\vec{c'} = \begin{pmatrix} c_x \\ c_y \\ \rho \end{pmatrix} \; ; \; \vec{r} = \begin{pmatrix} r_x \\ r_y \end{pmatrix} \; ; \; \vec{v'} = \begin{pmatrix} v_x \\ v_y \\ v_\rho \end{pmatrix}$$

On note X la combinaison de ces trois vecteurs :

$$X = \begin{pmatrix} \vec{c'} \\ \vec{r} \\ \vec{v'} \end{pmatrix}$$

4.2.3 Modèle de transition

Le modèle de transition correspond au modèle dynamique du système, il permet de prévoir l'état du système à un instant donné en fonction de son état précédent. Dans notre cas, l'état du système est caractérisé par la

position et la vitesse du segment. Si on considère une accélération nulle et aucune perturbation des segments, on obtient le modèle suivant :

$$X_{t+\Delta t} = \begin{pmatrix} \vec{c'}_{t+\Delta t} \\ \vec{r}_{t+\Delta t} \\ \vec{v'}_{t+\Delta t} \end{pmatrix} = \begin{pmatrix} \vec{c'}_t + \Delta t \cdot \vec{v'}_t \\ \vec{r}_t \\ \vec{v'}_t \end{pmatrix} \tag{4.9}$$

Cependant, un modèle plus évolué doit être considéré pour prendre en compte les perturbations agissant sur le système. Pour la position et la vitesse du centre, nous utiliserons un modèle d'accélération gaussien. A chaque itération, l'accélération est donc modélisée par une variable aléatoire $\vec{a}$ suivant une loi normale centrée en zéro. Pour la variation du demi-segment, nous considérons que le demi-segment subit une perturbation en angle suivant une loi normale centrée en zero et une perturbation en longueur proportionelle à sa taille.

Le modèle de transition devient donc :

$$X_{t+\Delta t} = \begin{pmatrix} \vec{c'}_{t+\Delta t} \\ \vec{r}_{t+\Delta t} \\ \vec{v'}_{t+\Delta t} \end{pmatrix} = \begin{pmatrix} \vec{c'}_t + \Delta t \cdot \vec{v'}_t \\ M_{\Delta t} \cdot \vec{r}_t \\ \vec{v'}_t + \Delta t \cdot \vec{a} \end{pmatrix} \tag{4.10}$$

où $\vec{a} \sim \mathcal{N}(0, \Sigma_1)$ est une variable aléatoire suivant une loi normale sur $\mathbb{R}^3$ de moyenne nulle et de covariance Σ_1, et $M_{\Delta t}$ est une matrice de rotation et changement d'échelle :

$$M_{\Delta t} = \begin{bmatrix} s^{\Delta t} & 0 \\ 0 & s^{\Delta t} \end{bmatrix} \cdot \begin{bmatrix} \cos(\Delta t \cdot \alpha) & -\sin(\Delta t \cdot \alpha) \\ \sin(\Delta t \cdot \alpha) & \cos(\Delta t \cdot \alpha) \end{bmatrix} \tag{4.11}$$

où s est une variable aléatoire suivant une loi normale centrée en 1 et d'écart type σ_2, et α est une variable aléatoire suivant une loi normale centrée en 0 et d'écart type σ_3.

4.2.4 Modèle d'observation

Comme on l'a vu précédement, le modèle d'observation représente la probabilité de l'observation connaissant l'état du système $P(Z_i|X_i)$. La première difficulté consiste à définir quelles sont les variables importantes dans notre système.

Une première possibilité est de considérer les segments détectés dans l'image de crêtes issus d'une première phase de détection (voir 3.4.3) comme des observations. Dans ce cas, on peut utiliser la distance avec l'objet détecté le plus proche pour évaluer la probabilité d'observation. Cependant, le suivi devient alors dépendant de la détection, et des erreurs de détection, telles

que des fausses détections ou des non détections peuvent alors provoquer des erreurs de suivi.

Ainsi, la solution que nous avons choisie consiste à utiliser deux mesures que nous fusionnerons. La première mesure est fondée sur la notion de segment de crête et utilisera la même fonction de score que celle utilisée pour la détection des structures. La seconde mesure est basée sur l'apparence visuelle et renvoie une probabilité élevée si le segment de crête observé ressemble au segment de crête prédit.

Modèle d'observation fondé sur la mesure de segment de crête

La première solution consiste à utiliser la même mesure que celle utilisée pour détecter les crêtes mais sans passer par la phase de seuillage et détection des maxima locaux. Ce modèle permet d'assurer que la cible suivie correspond toujours à un segment de crête tel qu'on l'a défini au chapitre précédent.

Le modèle d'observation est donc caractérisé par l'équation suivante :

$$P(Z_t|X_t) \propto \max(f_3(c_t, r_t), 0) \tag{4.12}$$

où f_3 est la fonction définie dans l'Equation 3.27.

Remarque 1 *La valeur* $p = \max(f_3(c_t, r_t), 0)$ *n'est pas une probabilité à proprement parler car la fonction* f_3 *n'est pas bornée. Cependant, il est possible de calculer une constante de normalisation en sommant les valeurs calculées sur toutes les particules. C'est pour cela que nous utilisons une relation de proportionalité* ($\propto$).

Modèle d'observation fondé sur la similarité visuelle

Lorsque la cible suivie est entourée d'autres segments de crêtes relativement proches, il peut être nécessaire de rajouter une contrainte permettant de discriminer les différentes cibles. Nous allons donc ajouter un descripteur à chaque cible suivie auquel nous associerons une mesure de similarité.

Descripteur associé aux segments de crêtes : Le descripteur que nous utilisons représente le profil de couleur le long de la crête. Il est codé par un tableau de taille $N_d + 1$ où chaque case i ($i \in [\![0, N_d]\!]$) contient une couleur $C(i)$ (au format RGB $\rightarrow$ vecteur de dimension 3) correspondant à un point du segment :

$$desc(\vec{c}, \vec{r}) = \{C_{\vec{c},\vec{r}}(i)\}_{i=0..N_d} \tag{4.13}$$

avec

$$C_{\vec{c},\vec{r}}(i) = L\left(\vec{c} + \left(\frac{2i}{N_d} - 1\right) \cdot \vec{r}\right) \tag{4.14}$$

où $L : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ est l'espace d'échelle gaussien sur les trois composantes rouge, vert et bleu ; et $\vec{c}$, $\vec{r}$ sont les paramètres (centre et demi-segment) du segment de crête.

Il faut noter que ce descripteur n'est pas robuste au changement d'illumination (en effet, un changement de luminosité modifie toutes les valeurs du descripteur). Cela peut causer des problèmes par exemple lorsque l'objet suivi entre ou sort d'une zone d'ombre ou lorsque les conditions d'éclairages changent (passage d'un nuage, prise d'image en contre-jour, etc.). Nous aurions pu choisir d'autres types de descripteurs plus robustes à ces variations, tel que le descripteur SIFT [Low99, Low04] fondé sur des histogrammes du gradient. Cependant, notre descripteur est intéressant pour plusieurs raisons :

- Le coût de calcul est très faible.
- Le descripteur ne prend en considération que des pixels situés sur l'objet, il n'est donc pas perturbé par l'apparence du fond. Avec le descripteur SIFT, le gradient à la frontière de la zone d'intérêt dépend fortement de la couleur de fond.
- Le descripteur utilise l'information de couleur (3 dimensions), ce qui permet une large discrimination entre les différents objets.

Initialisation et mise à jour du descripteur : Le descripteur est initialisé en même temps que l'initialisation du filtre en considérant l'état initial du filtre. Cependant, l'apparence des objets peut varier légérement au cours du temps, il est donc nécessaire de mettre à jour le descripteur. Pour cela, à chaque itération, nous pouvons extraire du filtre un segment représentant au mieux l'état du filtre (moyenne des particules ou la meilleure particule), nous calculons le descripteur associé à ce segment de crête et nous l'intégrons au descripteur de référence à l'aide d'un filtre passe bas :

$$\forall i \in [\![0, N_d]\!], C^t_{ref}(i) = (1 - \alpha)C^{t-1}_{ref}(i) + \alpha C_t(i) \tag{4.15}$$

où $C^t_{ref}(i)$ et $C^{t-\Delta_t}_{ref}(i)$ correspondent aux cases du descripteur de référence aux instants t et $t - \Delta_t$, $C_t(i)$ correspondent aux cases du decripteur associé au segment de crête estimé par le filtre. α est une constante comprise entre 0 et 1 représentant l'oubli : une valeur de 0 est équivalente à aucune mise à

jour (aucun oubli du descripteur de référence) et une valeur de 1 signifie que le descripteur est réinitialisé à chaque itération (oubli total du descripteur de référence).

Mesure de similarité entre deux descripteurs : Pour comparer deux descripteurs nous avons besoin d'une mesure de similarité entre deux blocs de pixels. Un certain nombre de méthodes sont courament utilisées : SAD (Sum of Absolute Differences), SSD (Sum of Square Differences), ZNCC (Zero mean Normalised Cross Correlation), etc. La ZNCC consiste à comparer les deux blocs après avoir normalisé chacun des bloc par leur moyenne. Ainsi seule la variation de couleur par rapport à la moyenne joue un rôle. Cette mesure ne donnera pas de bon résultats dans notre cas puisque la couleur des segments suivi varie relativement peu le long du segment. La SAD et la SSD correspondent respectivement à la norme 1 et la norme 2 du vecteur associé à la différence des deux blocs. Les résultats entre les deux mesures sont généralement assez proches, nous avons donc choisi arbitrairement d'utiliser la méthode SSD :

$$SSD(desc_1, desc_2) = \sum_{i=0}^{N_d} \|C_1(i) - C_2(i)\|^2 \tag{4.16}$$

La distance associée à cette mesure est la suivante

$$dist(desc_1, desc_2) = \sqrt{\frac{1}{N_d} \sum_{i=0}^{N_d} \|C_1(i) - C_2(i)\|^2} \tag{4.17}$$

Modèle d'observation : Avant de fusioner l'information de similarité, il est nécessaire de transformer cette mesure en probabilité. Pour cela, on va assumer que la distance entre le descripteur $desc_{X_t}$ de la cible suivie et le descripteur observé $desc_{Z_t}$ suit une loi normale :

$$P(Z_t|X_t) \propto e^{-\frac{dist(desc_{Z_t}, desc_{X_t})^2}{2\sigma^2}} \tag{4.18}$$

En pratique, si les valeurs d'intensité de chaque couleur sont comprisent entre 0 et 1, la distance entre 2 descripteurs est aussi comprise entre 0 et 1. La valeur de sigma que nous avons choisi est de l'ordre de 0.1.

Fusion des deux modalités

Le principe de la fusion de données est d'utiliser plusieurs observations de nature différente pour estimer une variable X.

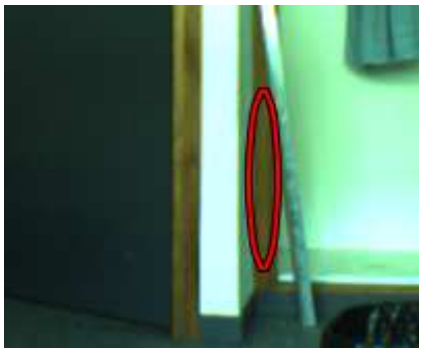
Dans notre cas de filtrage bayésien, le but de la fusion est d'évaluer la probabilité des différentes observations conditionellement à l'état du filtre :

$$P(Z_t^1 Z_t^2 | X_t)$$

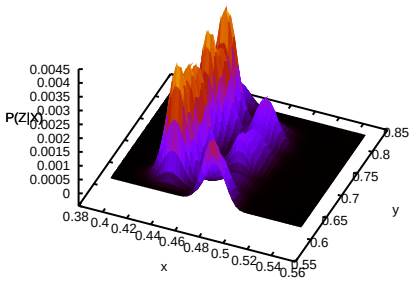
où Z_t^1 et Z_t^2 sont les deux observations liées respectivement à la mesure de segment de crête et à la similarité visuelle. Afin de simplifier grandement la tâche, nous allons faire l'hypothèse que les deux observations sont indépendantes connaissant la variable X_t. Cette hypothèse est raisonable du fait que les deux observations utilisent des mesures de natures différentes (contraste et couleur). Le modèle d'observation peut alors s'exprimer comme le produit des deux modèles d'observations :

$$P(Z_t^1 Z_t^2 | X_t) = P(Z_t^1 | X_t) P(Z_t^2 | X_t) \tag{4.19}$$

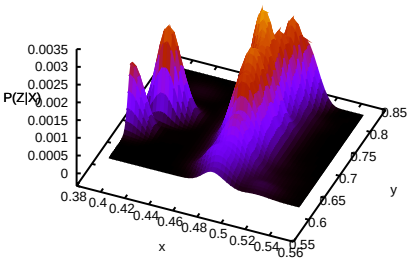
La Figure 4.2 montre le résultat de ces deux modèles d'observation et de la fusion de ces deux modèles. On peut remarquer que les deux modèles ont une distribution multi-modale dans le cas illustré et peuvent donc causer des erreurs de suivi si ils sont utilisés seuls ; par contre la fusion des deux modèles à une distribution unimodale qui est bien localisée sur la cible considérée.



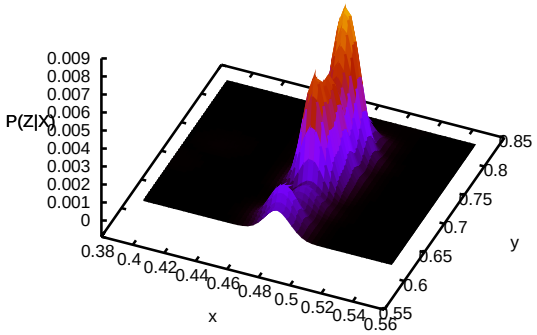
(a) Image de référence : le modèle capteur est évalué à l'emplacement du segment de crête représenté par une ellipse et en considérant uniquement les variations en (x, y).



(b) Modèle d'observation fondé sur la mesure de crête. On peut observer deux modes principaux qui correspondent au segment de crête observé (à droite) et à la zone blanche et qui est plus marquée que le premier : si on considère uniquement ce modèle d'observation on va donc converger vers une autre cible.



(c) Modèle d'observation fondé sur la similarité des descripteurs. Ici aussi on observe deux modes, celui de gauche correspondant à la seconde zone foncée à gauche de la cible, mais la zone blanche n'apparaît plus.



(d) Modèle d'observation résultant de la fusion des deux modèles. Le résultat de la fusion n'est composé que d'un seul pic principal localisé sur la cible suivie.

FIG. 4.2 – Résultat des deux modèles d'observation et de la fusion des deux.

4.2.5 Suivi multi-cibles

Lorsque l'on suit plusieurs cibles simultanément, la gestion des différentes cibles est un problème primordial. En effet, les cibles peuvent disparaître (sortie du champ de vision, masquage par d'autres objets, changement d'apparence ne permettant plus la détection, etc.), plusieurs cibles peuvent se regrouper (par exemple si deux objets se rapprochent jusqu'à ne former plus qu'un seul objet détectable), et des nouvelles cibles apparaissent sans cesse. Il est donc nécessaire de gérer tout ces cas afin de suivre en permanence le maximum de cibles possibles et en supprimant les cibles qui disparaissent.

Nous avons choisi d'utiliser une méthode relativement simple qui consiste à suivre en permanence un nombre constant de cibles de manière indépendante, puis à chaque itération, nous détectons les cas d'erreur (sortie du champ de vue, fin de détection, regroupement avec une autre cible, etc.) et dans ces cas, la cible est réinitialisée en choisissant aléatoirement une cible parmi les objets renvoyés par le détecteur.

Détection des cas d'erreurs

On distingue deux cas d'erreurs de nature différente : la disparition d'une cible et le regroupement de deux cibles.

Pour détecter le premier cas, nous évaluons la probabilité de l'observation en utilisant la Formule 4.19 : si toutes les particules ont une probabilité d'observation inférieure à un certain seuil, on considère alors que la cible suivie a disparu. Ce critère permet aussi de supprimer les cibles qui sortent du champ de vue car nous considérons comme nulle la probabilité d'observation d'une particule qui est en dehors du champ de l'image.

Pour détecter les regroupements, nous évaluons les distances entre les positions moyennes de chaque cible suivie. Si cette distance est faible, nous considérons que les deux cibles se sont regroupées, il faut alors choisir une des deux cibles et la supprimer, il est possible de supprimer n'importe quelle des deux cibles, mais il est plus judicieux de supprimer la cible la moins âgée. De même seul le descripteur associé à la cible la plus âgée est conservé.

Le choix de la distance entre deux segments de crête n'est pas évident, la distance doit en effet tenir compte de la distance entre les centres mais aussi de la différence de taille et d'orientation. Pour définir cette distance, nous considérons un segment de crête comme une ellipse dont le centre correspond au centre du segment de crête, le grand axe correspond au demi-segment et la longueur du second axe est déterminée par l'échelle de la crête (voir Figure 4.3). Ainsi, chaque segment de crête est maintenant caractérisé par le centre de l'ellipse $\vec{p}$ en deux dimensions et la matrice de covariance Σ associée

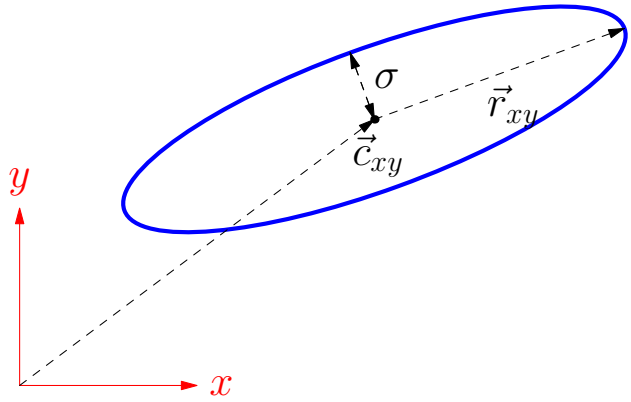


FIG. 4.3 – Représentation d'un segment de crête par une ellipse.

à l'ellipse.

Pour comparer deux crêtes, on peut alors faire le lien entre les ellipses et les distributions gaussiennes, nous allons donc naturellement comparer les deux ellipses correspondant aux segments de crêtes de la même manière que l'on compare deux distributions de probabilité, c'est-à-dire en calculant l'entropie croisée (ou divergence de Kullback-Leiber) des distributions gaussiennes associées aux deux ellipses.

Définition 8 *Divergence de Kullback-Leibler :*

Pour deux distribution de probabilités p et q de la variable x dans $\mathbb{R}^N$, la divergence de Kullback-Leibler de p et q est définie par :

$$D_{KL}(p\|q) = \int_{\mathbb{R}^N} \log\left(\frac{p(x)}{q(x)}\right) p(x)dx$$

Théorème 1 *La divergence de Kullback-Leiber entre deux distributions normales p_1 et p_2 de moyennes respectives μ_1, μ_2 et de covariances respectives Σ_1 et Σ_2 est :*

$$D_{KL}(p_1\|p_2) = \frac{1}{2}\left[\log\left(\frac{det(\Sigma_2)}{det(\Sigma_1)}\right) + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2) - N\right]$$

où det et tr sont respectivement le déterminant et la trace et N est la dimensionalité de l'espace.

Remarque 2

- *Cette divergence fait intervenir trois termes : les deux premiers termes tiennent uniquement compte des matrices de covariances et caractérisent donc la dissimilarité due aux tailles et aux orientations des deux ellipses, le second terme correspond à la distance de Mahalanobis entre les deux centres.*

– *La divergence n'est pas une distance (non symétrique et inégalité triangulaire non respectée), cependant il est possible d'utiliser une forme symétrique :*

$$D(p_1, p_2) = D_{KL}(p_1 \| p_2) + D_{KL}(p_2 \| p_1) \tag{4.20}$$

– *La divergence est nulle si* $\Sigma_1 = \Sigma_2$ *et si* $\mu_1 = \mu_2$*, c'est-à-dire si les deux segments sont identiques.*

Ainsi, pour décider si deux segments de crêtes S_1 et S_2 correspondent à la même cible nous calculons la divergence (sous la forme symétrique) entre les deux formes elliptiques correspondant aux deux segments. Plus cette valeur est faible, plus les deux segments sont proches. Dans la suite, nous noterons $\varphi(S_1, S_2)$ cette valeur. La Figure 4.4 donne une visualisation de cette distance pour différentes paires d'ellipse. On peut remarquer que lorsque la divergence est inférieure à 1, les deux ellipses comparées sont visuellement très proches.

Avec cette méthode, il y a en permanence un nombre constant de filtres à particules, ce qui permet d'avoir des temps de calculs constants et donc

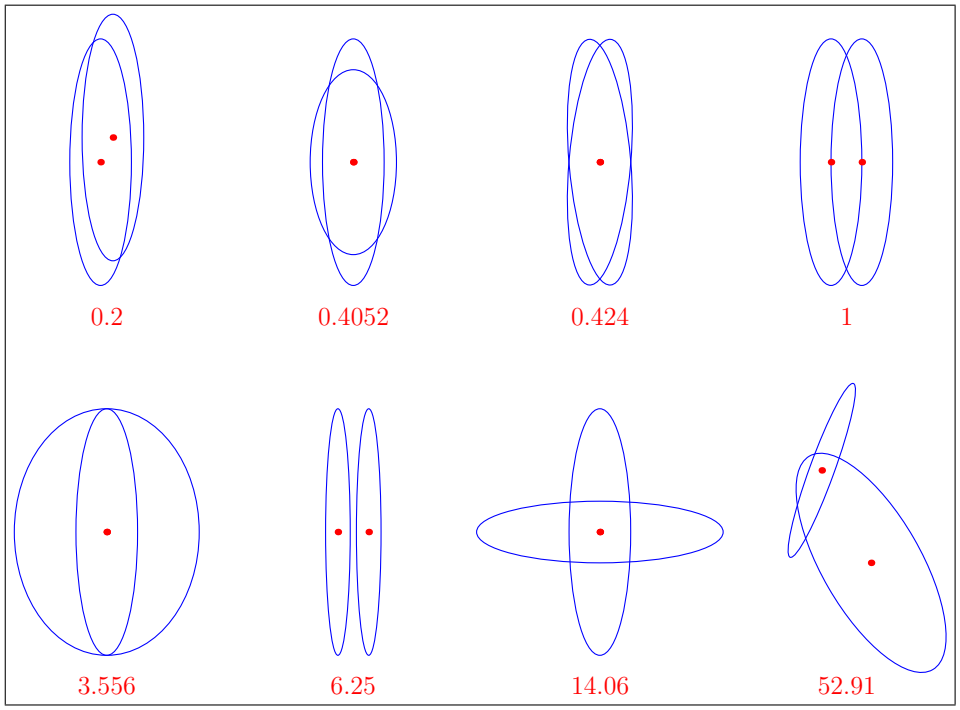


FIG. 4.4 – Divergence de Kullback-Leiber (forme symétrique) pour différentes paires d'ellipses. Lorsque la mesure est inférieure à 1, les deux ellipses sont visuellement proches.

prévisibles, ce qui est bien adapté au temps réel. Par contre, il est possible que certaines cibles soient bien détectées mais ne soient pas suivies si tous les filtres suivent déjà des cibles différentes, il faut donc prévoir un nombre important de filtres à particules pour ne pas être dans ce cas.

4.2.6 Algorithme global

Pour récapituler, la méthode globale du suivi des segments de crête est décrite par l'Algorithme 5. On peut noter que le coût global pour chaque itération avec une nouvelle image est de l'ordre du nombre de filtres utilisés multiplié par le nombre de particules. De plus, cet algorithme est bien parallélisable puisque chaque filtre fonctionne de manière indépendante.

4.3 Conclusion

Dans ce chapitre, nous avons étudié une méthode de suivi des structures définies dans le chapitre précédent. Ce suivi utilise le concept de filtre bayésien que nous avons détaillé au début du chapitre, et plus précisément de filtre à particules. Nous avons ainsi étudié un modèle dynamique du déplacement des segments de crête dans l'espace d'échelle correspondant au modèle de transition du filtre bayésien.

Ensuite, nous avons mis au point un modèle d'observation adapté à de telles cibles. L'originalité de ce modèle est d'utiliser non pas la sortie du détecteur de segment de cible mais la fonction de détection elle même. Cela évite d'avoir à fixer un seuil de détection et d'effectuer une recherche de maximum. On évite aussi une phase d'association de données délicate. Pour avoir une meilleure fiabilité, le modèle d'observation fondé sur la mesure de laplacien est fusionné avec une seconde modalité utilisant la couleur des pixels le long du segment. Enfin, nous avons étudié une méthode permettant de suivre simultanément plusieurs cibles différentes en détectant les pertes et les fusions entre différentes cibles.

Les performances de cet algorithme de suivi serons analysées dans le Chapitre 6.2.2. On verra que le suivi est très robuste aux changements d'échelles, ce qui est notre principal objectif. Ensuite, le suivi multi-cibles fonctionne bien lorsque les éléments sont bien distincts. Cependant, lorsque l'environnement est très encombré, il devient difficile de suivre séparément chaque objet, surtout dans les petites échelles où les vitesses relatives des éléments par rapport à leur taille est grande.

Dans le chapitre suivant, nous allons voir comment utiliser le résultat de cet algorithme pour évaluer les risques de collision associés à chaque cible.

Algorithme 5 Algorithme de suivi des segments de crête

1. Récupérer la première image caméra ;
2. Détecter les segments de crêtes ;
3. Initialiser tous les filtres avec un segment de crête choisi aléatoirement ;
4. **pour** chaque filtre i : $a_i \leftarrow 0$ **fin pour**
5. **boucler**
   (a) Récupérer la nouvelle image caméra ;
   (b) Détecter des nouveaux segments de crêtes ;
   (c) **pour** chaque filtre i :
      i. Application du modèle de transition pour chaque particule ;
      ii. Calcul du poids de chaque particule en appliquant et en fusionant les deux modèles d'observation ;
      iii. Recherche de la meilleure particule $\overline{S_i}$ et de sa probabilité d'observation $\overline{p_i}$;
      iv. **si** $p_i < seuil$ ou **si** $\exists j / (\varphi(\overline{S_i}, \overline{S_j}) < \epsilon \; et \; a_i \leq a_j)$ **alors**
         A. Réinitialiser le filtre en choisissant aléatoirement un segment de crête détecté en 5b ;
         B. $a_i \leftarrow 0$;

         sinon

         A. Appliquer le rééchantillonage du filtre si nécessaire et normaliser le poids des particules ;
         B. $a_i \leftarrow a_i + 1$;
   (d) **fin pour**
6. **fin boucle**

Chapitre 5

Evaluation des risques de collision

5.1 Utilisation du temps avant collision

5.1.1 Définition

Le temps avant collision, souvent noté TTC pour *Time-To-Collision*, *Time-To-Contact* ou encore *Time-To-Crash* en anglais a été introduit dans le millieu des année 70 par Lee [Lee76]. Le TTC est une mesure temporelle de l'espace compris entre un observateur et un autre objet. Il est défini comme le temps que mettrait un observateur pour entrer en contact avec un objet vers lequel il se dirige. Bien sûr, cette mesure n'est valable qu'à un instant donné et si la vitesse relative entre l'observateur et l'objet considéré reste constante.

De manière formelle, le temps avant collision, que nous noterons τ par la suite, est le rapport entre la distance observateur/objet Z et la vitesse de rapprochement $-\dot{Z}$:

$$\tau = -\frac{Z}{\dot{Z}} \tag{5.1}$$

Toutefois, comme le montre Lee dans [Lee81] et [LYR92], l'intérêt de cette mesure réside dans le fait qu'elle peut être évaluée de manière visuelle sans mesurer directement la distance de l'objet considéré. On peut observer en effet que certains êtres vivants utilisent le TTC, pour des manoeuvres d'évitement ou de chasse par exemple, sans estimer séparément la vitesse de rapprochement ni la distance à un objet.

5.1.2 Travaux antérieurs

Pour évaluer le temps avant collision, les travaux antérieurs sont essentiellement fondés sur le flux optique. Ce flux optique est défini comme le mouvement de l'image sur la rétine (humaine ou artificielle), il représente donc la projection du champ des vitesses de l'espace sur la rétine. Le concept de base utilisé pour évaluer le TTC est le fait que lorsque un objet approche, les pixels qui le représentent dans l'image s'écartent. Camus [Cam94] calcule le centre d'expansion du flux optique, le temps avant collision est alors calculé pour chaque pixel en calculant le rapport entre le flux optique et la distance au centre d'expansion. Cependant, cette méthode ne fonctionne plus lorsque plusieurs objets bougent de manière indépendante, chaque objet ayant alors un centre d'expansion différent. Dans [CCHH96] et [CHHN98], une expression plus locale du TTC à été définie, on y trouve ainsi la formulation $div\vec{f} \approx \frac{2}{\tau}$ où div est l'opérateur de la divergence et $\vec{f}$ est le champ de vecteurs du flux optique. Les auteurs ont équipé un robot d'une simple caméra. En utilisant cette approximation, le temps avant collision est calculé en temps réel sur chaque partie de l'image. Le robot se dirige en permanence vers les zones ou le temps avant collision est le plus grand, ce qui minimise les risques de collision avec un obstacle.

Cependant, ces méthodes nécessitent le calcul du flux optique de manière dense et héritent donc des problèmes et des faiblesses des algorithmes de calculs du flux optique, à savoir par exemple des temps de calculs prohibitifs pour obtenir une bonne précision et une haute résolution. De plus, les algorithmes de flux optique ne donnent de bons résultats que lorsque l'image est bien texturée, c'est-à-dire en général uniquement sur les contours des objets. Lorsque la scène contient des objets relativement larges et faiblement texturés il faut utiliser des méthodes complexes de calculs fondées sur des méthodes d'optimisation [BWF+05, BWS05].

Dans la suite de ce chapitre nous proposons de calculer le temps avant collision en utilisant le changement d'échelle des objets dans l'image. Ce changement d'échelle peut être obtenu à partir d'un système de suivi quelconque fournissant la taille des éléments suivis dans l'image. Dans notre cas, nous allons utiliser la sortie de l'algorithme de suivi présenté dans le chapitre précédant fournissant entre autre l'échelle caractéristique des objets (au sens de l'espace d'échelle) à tout instant.

5.2 Evaluation du temps avant collision par variation d'échelle

5.2.1 Modèle caméra utilisé

Nous utilisons un modèle projectif de caméra, plus connu sous le nom *pinhole* en anglais signifiant trou-d'épingle. Ce modèle est illustré par la Figure 5.1 et consiste à projeter les points de l'espace 3D sur un plan grâce à une projection centrale. Ce modèle est très utilisé car il permet d'effectuer tous les calculs sous forme linéaire (en utilisant les coordonnées homogènes) et car il modélise très bien les caméras classiques (à condition que l'angle de vision soit relativement faible).

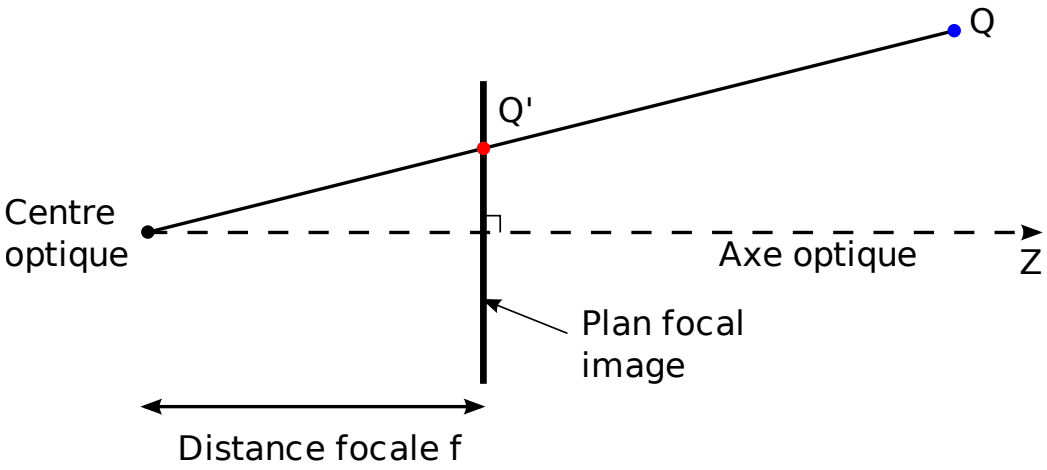


FIG. 5.1 – Modèle caméra utilisé. Le point 3D Q est projeté sur le plan image par projection centrale à partir du centre optique.

Avec une telle représentation, la caméra est caractérisée par 3 paramètres[1] : (u_0, v_0) qui représentent les coordonnées du point principal (position de l'intersection entre l'axe optique et le plan image, dans le repère image), et f qui représente la distance focale dans l'unité pixel. On obtient

[1]En général on utilise 4 paramètres en séparant la distance focale en x et en y [HZ00], ce qui permet de prendre en compte l'aspect des pixels. Toutefois, pour une caméra numérique, on peut généralement supposer que les pixels sont carrés et utiliser la même valeur pour les deux dimensions.

ainsi la matrice de projection P :

$$P = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

La projection Q' d'un point 3D Q sur le plan image est alors obtenue par simple produit avec la matrice de projection :

$$Q' = PQ = P \begin{bmatrix} Q_x \\ Q_y \\ Q_z \end{bmatrix}$$

où Q' est exprimé en coordonnées homogènes, on a donc en coordonnées standard :

$$q' = \begin{bmatrix} \frac{Q'_x}{Q'_z} \\ \frac{Q'_y}{Q'_z} \end{bmatrix} = \begin{bmatrix} \frac{fQ_x}{Q_z} + u_0 \\ \frac{fQ_y}{Q_z} + v_0 \end{bmatrix}$$

En pratique, ce modèle n'est pas totalement respecté, surtout lorsque le champ de vision est grand, pour appliquer ce modèle il faut donc déformer préalablement l'image en appliquant un modèle de distorsion supplémentaire.

5.2.2 Relation entre l'échelle et le temps avant collision

On peut remarquer facilement que lorsqu'un objet s'approche d'une caméra, la taille de cet objet observé dans l'image augmente, le lien entre la distance d'un objet et sa taille dans l'image (ou son échelle) est donc évident. Le temps avant collision étant fonction de la distance et du temps, il existe donc une relation entre l'échelle et le temps avant collision. Nous allons donc formaliser mathématiquement cette relation.

Considérons une caméra fixe et un objet rigide se déplaçant avec un mouvement de translation pure (voir Figure 5.2).

Soit $Q_1 = [\ x_1 \quad y_1 \quad z_1\ ]^T$ et $Q_2 = [\ x_2 \quad y_2 \quad z_2\ ]^t$ deux points appartenant à l'objet, exprimés dans le repère caméra.

On note $\vec{r} = [\ r_x \quad r_y\ ]^t$ le vecteur entre les deux points projetés dans l'image, et $\vec{R} = (R_x, R_y, R_z)^t$ le vecteur entre les deux points dans l'espace 3D, on a alors :

$$\vec{r} = \begin{bmatrix} f \cdot (\frac{x_1}{z_1} - \frac{x_2}{z_2}) \\ f \cdot (\frac{y_1}{z_1} - \frac{y_2}{z_2}) \end{bmatrix}$$

On fait alors l'hypothèse que : $\|z_1 - z_2\| \ll z_1$ (et donc $z_1 \approx z_2 \approx z$)

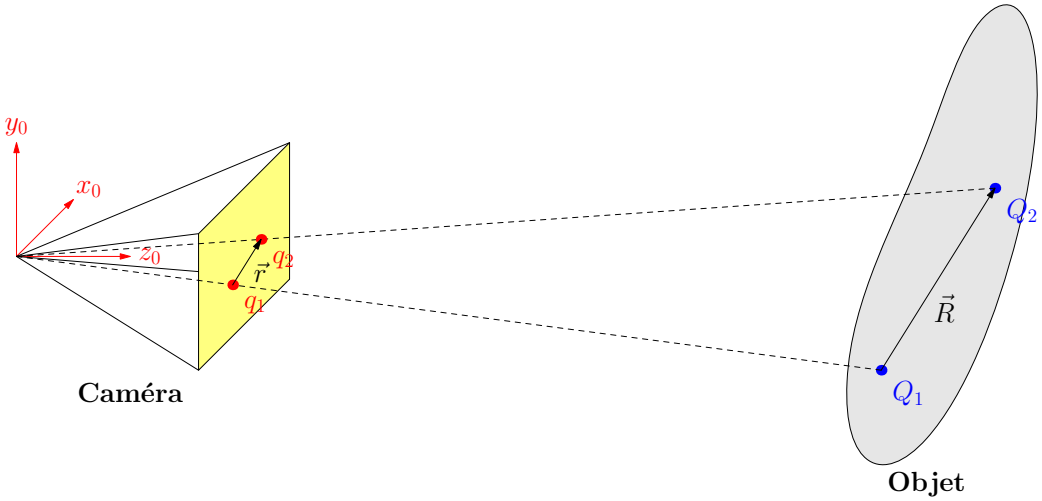


FIG. 5.2 – Représentation de la caméra et d'un objet visible. Les points Q_1 et Q_2 appartenant à l'objet sont projetés dans l'image sur les points q_1 et q_2.

On peut alors faire l'approximation :

$$\vec{r} \approx \frac{1}{z} \begin{bmatrix} f \cdot R_x \\ f \cdot R_y \end{bmatrix} \tag{5.2}$$

d'où

$$z \approx \frac{f \cdot R_x}{r_x} \approx \frac{f \cdot R_y}{r_y} \tag{5.3}$$

et donc :

$$\frac{dz}{dt} \approx -\frac{f \cdot R_x}{r_x^2} \cdot \frac{dr_x}{dt} \tag{5.4}$$

Le temps avant collision peut alors s'exprimer par rapport à $\vec{r}$:

$$\tau = -\frac{z}{\frac{dz}{dt}} \approx \frac{r_x}{\frac{dr_x}{dt}} \approx \frac{r_y}{\frac{dr_y}{dt}} \tag{5.5}$$

On a vu dans la partie 3.2.4 qu'il est possible, pour un objet observé dans l'image, d'extraire une échelle caratéristique σ à partir de l'espace d'échelle laplacien. Cette échelle correspondant à une taille caractéristique d'un objet dans l'image, on peut donc remplacer r par σ et on obtient :

$$\tau \approx \frac{\sigma}{\frac{d\sigma}{dt}} \tag{5.6}$$

Ainsi, grâce à cette équation, si on est capable d'estimer l'échelle d'un objet dans une image à tout instant, il est alors possible d'estimer son temps avant collision. On peut noter que les paramètres de la caméra n'interviennent plus dans cette équation, il est donc possible de travailler avec une caméra non calibrée.

La Figure 5.3 illustre deux courbes du temps avant collision évalué à partir de l'évolution de l'échelle caractéristique. On peut noter que lorsque l'échelle diminue, le TTC est négatif, ce qui signifie que l'obstacle s'éloigne.

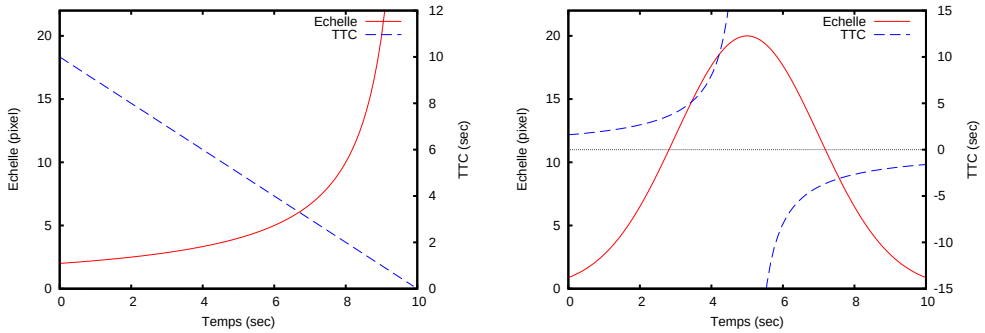


FIG. 5.3 – Deux exemples de courbe d'échelle et du temps avant collision correspondant.

5.2.3 Calcul du temps avant collision

Dans les deux chapitres précédant, nous avons vu qu'il est possible de suivre des objets particuliers dans une image. Ce suivi s'effectuant dans l'espace d'échelle, nous pouvons avoir une échelle caractéristique de l'objet suivi à tout instant (en plus de sa position et sa vitesse dans l'image), on peut donc alors calculer la variation de cette échelle et utiliser la Formule 5.6 pour en déduire le temps avant collision. Toutefois, les mesures d'échelles peuvent être significativement bruitées et un calcul de dérivée par simple différentiation va donc être très instable. Il est donc nécessaire de filtrer une seconde fois les données d'échelles provenant de la phase de détection et de suivi de cibles.

D'après la définition du temps avant collision, on considère que la vitesse relative de l'obstacle par rapport à la caméra est constante. Cela signifie que la distance relative entre ces deux éléments varie linéairement en fonction du temps :

$$z(t) = a \cdot t + b \tag{5.7}$$

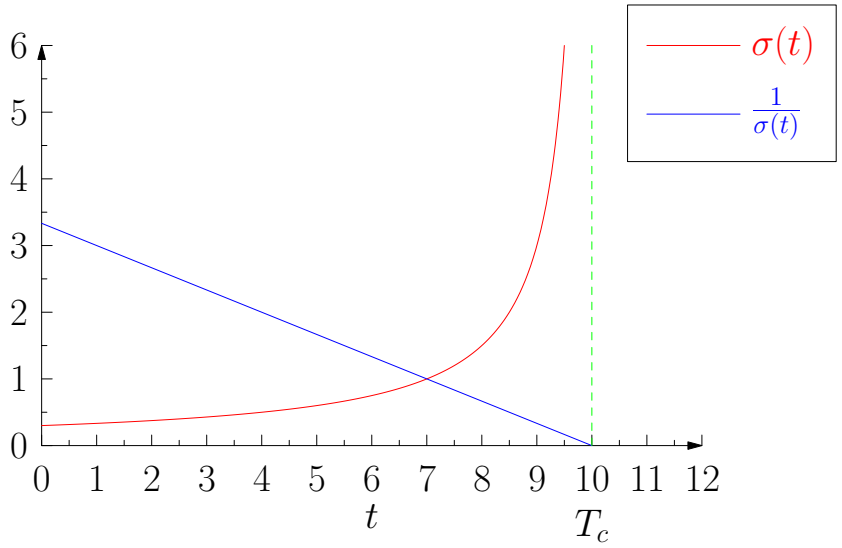


FIG. 5.4 – Evolution de l'échelle ($\sigma(t)$) et de l'inverse de l'échelle d'un obstacle en fonction du temps lorsque la caméra se déplace à vitesse constante par rapport à cet obstacle. L'inverse de l'échelle évolue de manière linéaire et la collision à lieu lorsque cette valeur vaut 0.

La collision a lieu lorsque $z = 0$, c'est-à-dire :

$$\tau = -\frac{b}{a} \tag{5.8}$$

En reprenant l'Equation 5.3 et en notant R la taille caractéristique de l'objet dans l'espace 3D et $\sigma(t)$ la taille caractéristique de l'objet dans l'image à l'instant t, on a :

$$z(t) \approx \frac{f \cdot R}{\sigma(t)} \tag{5.9}$$

et donc :

$$\frac{1}{\sigma(t)} = \frac{1}{f \cdot R}(a \cdot t + b) = a' \cdot t + b' \tag{5.10}$$

On peut donc estimer le temps avant collision en effectuant une régression linéaire de l'inverse de l'échelle par rapport au temps.

Cependant, si la vitesse relative de l'objet par rapport à la caméra varie au cours du temps, l'échelle ne suit plus totalement cette loi (la pente de la courbe va varier avec le temps). Pour être réactif et tenir compte rapidement de ces variations, il faut donc oublier progressivement les anciennes mesures. Pour cela, nous allons utiliser une méthode similaire à

[CHHN98] qui consiste à effectuer une résolution de moindres carrés récursifs avec un oubli exponentiel.

Dans notre cas, nous cherchons les paramètres $\vec{H}(t) = [\ a \quad b\ ]^T$ qui minimisent le résidu :

$$J(t) = \sum_{i=1}^{n} \lambda^{t-t_i} \left(\frac{1}{\sigma(t_i)} - (a \cdot t_i + b) \right)^2 \tag{5.11}$$

Où n représente le nombre de mesures, t_k (avec $t_k < t$) représente l'instant où a été effectué la mesure et $\lambda \in [0, 1]$ est le paramètre d'oubli (ou de mémoire). Si on fixe λ à 0, cela signifie qu'on ne tient compte que de la dernière mesure (mémoire nulle) et lorsque $\lambda = 1$, cela revient à donner un poids équivalent à toutes les mesures (mémoire infinie).

En notant $d_i = \frac{1}{\sigma(t_i)}$, $X_i = \left[\ t_i \quad 1\ \right]^T$ et $H_i = \left[\ a_i \quad b_i\ \right]^T$, on peut récrire l'Equation 5.11 par :

$$J(t) = \sum_{i=1}^{n} \lambda^{t-t_i} (d_i - H_i^T X_i)^2 \tag{5.12}$$

On minimise de manière récursive ce résidu en utilisant l'algorithme des moindres carrés récursifs (voir Algorithme 6).

Toutefois, nous devons modifier légèrement cet algorithme si les données ne sont pas espacées uniformément dans le temps. Si on note $\Delta t = t_i - t_{i-1}$ la durée entre deux mesures, il faut alors remplacer λ par $\lambda^{\Delta t}$ pour avoir un oubli exponentiel qui est vraiment fonction du temps et non pas de l'indice de la mesure.

Algorithme 6 Moindres carrés récursifs avec oubli exponentiel

Entrées:

– $X_1, X_2, \ldots, X_n$: signal d'entrée
– $d_1, d_2, \ldots, d_n$: signal observé

Sorties:

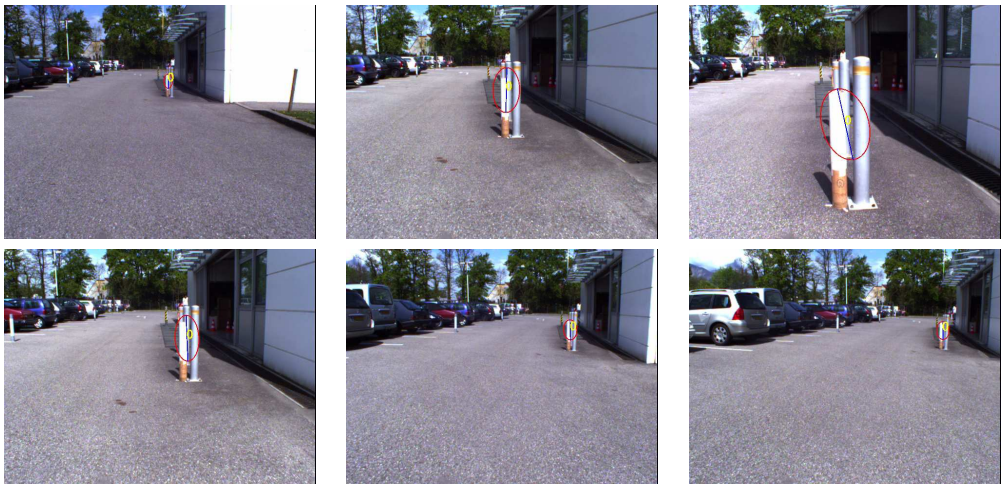
– $H_1, H_2, \ldots, H_n$: paramètres du modèle qui minimisent :

$$J(k) = \sum_{i=1}^{k} \lambda^{k-i}(d_i - H_k^T X_i^T)^2$$

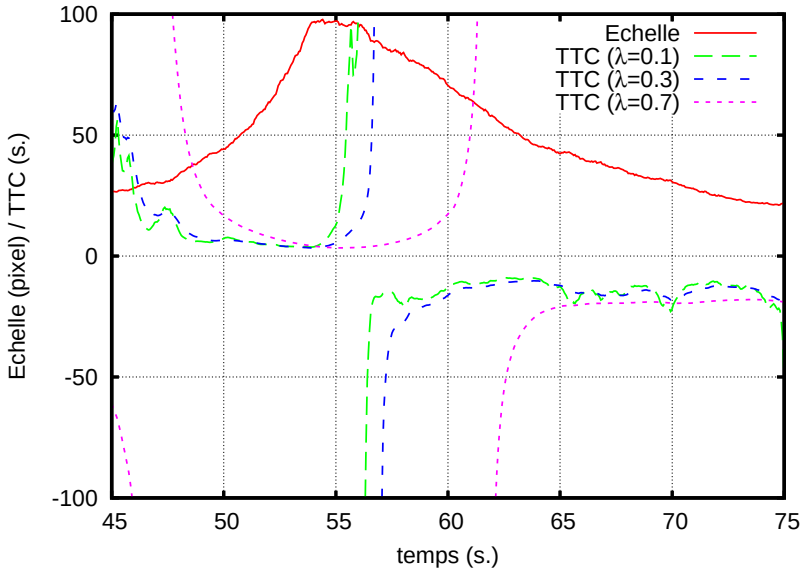
pour $k = 1..n$

Algorithme :

1. $Q_0 = \frac{I}{\delta}$, $H_0 = 0$ avec :
   – I : matrice identité
   – δ réel tel que $\delta > 0$ et $\delta \ll 1$
2. **pour** $i = 1..n$ **faire**
   (a) $K_i = \frac{Q_{i-1}X_i}{\lambda + X_i^T Q_{i-1} X_i}$
   (b) $H_i = H_{i-1} + K_i(d_i - X_i^T H_{i-1})$
   (c) $Q_i = \lambda^{-1} Q_{i-1} - \lambda^{-1} K_i X_i^T Q_{i-1}$
3. **fin pour**



(a) Séquence vidéo utilisée (de gauche à droite et de haut en bas). L'obstacle suivi est indiqué par l'ellipse rouge.



(b) TTC calculé en fonction du temps.

FIG. 5.5 – Séquence de test (a) et temps avant collision calculé à partir de l'échelle et en utilisant le filtre des moindres carrés récursifs avec différentes valeurs du paramètre λ (b). Au début, on se rapproche de l'obstacle, ce qui se traduit par une augmentation de l'échelle. Le temps avant contact diminue donc jusqu'à atteindre quasiment 0. Ensuite, la caméra s'éloigne de l'obstacle, ce qui se traduit par un TTC négatif. On peut observer l'effet du paramètre λ : avec un λ proche de zéro, on obtient une réponse assez bruitée mais avec très peu de retard. (temps de réaction faible). Lorsqu'on augmente λ la réponse est plus lissée, mais le retard est plus grand.

5.3 Conclusion

Dans ce chapitre nous avons proposé une nouvelle méthode d'estimation du temps avant collision utilisant uniquement la variation d'échelle caractéristique des objets dans l'image. Le temps avant collision ainsi calculé fournit une information précieuse sur les risques de collision liés à chaque obstacle. Cette mesure temporelle permet de prendre en compte la dynamique des obstacles et est donc mieux adaptée dans un environnement dynamique que des mesures de distances spatiales. De plus, cette méthode ne requiert aucune calibration préalable de la caméra utilisée. La précision de cette méthode dépend fortement de la qualité du suivi et plus particulièrement de l'estimation de l'échelle caractéristique des objets dans l'image. Nous avons toutefois proposé une méthode de filtrage permettant de diminuer l'effet du bruit et des imprécisions de mesure de l'échelle, mais en prennant néanmoins le risque d'introduire du retard dans l'estimation du TTC.

Chapitre 6

Réalisations et évaluation expérimentale

6.1 Architecture matérielle et logicielle

6.1.1 Plateforme expérimentale

Notre plateforme expérimentale se compose d'un robot mobile *Cycab* de type voiture (Figure 6.1), ayant quatre roues motrices et deux directions (avant et arrière). Le système informatique du Cycab comprend deux micro-contrôleurs chargés de commander la vitesse et l'angle de braquage de chaque roue, et d'un ordinateur embarqué permettant de commander ces deux micro-contrôleurs et de connecter différents capteurs (GPS, télémètre laser). Pour connecter les caméras, on dispose d'un PC portable supplémentaire muni d'un port firewire. Les deux PC sont reliés par éthernet afin de pouvoir communiquer. L'architecture de la plateforme est illustrée sur la Figure 6.2.

6.1.2 Cycabtk

Pour réaliser les expérimentations sur la plateforme Cycab, nous avons mis au point un ensemble d'outils permettant d'accéder de manière simulée ou réelle aux différentes ressources de la plateforme (envoi de commande en vitesse, acquisition des capteurs, etc.).

Ce logiciel a été conçu de manière à répondre à différents besoins particulièrement importants pour effectuer des expérimentations en robotique :

Abstraction du matériel : une couche d'abstraction permet d'accéder de la même manière à différents capteurs ou actionneurs, du moment

FIG. 6.1 – La plateforme expérimentale *Cycab*.

que les données sont du même type (images caméras, commandes en vitesse, position GPS, etc.). Ainsi, on peut utiliser exactement la même application (sans même avoir à recompiler le code) avec des données réelles ou simulées, ou en utilisant des capteurs de différents constructeurs. Cette couche d'abstraction est implantée sous forme de mémoire partagée. Chaque composant peut ainsi lire ou écrire des variables dans un espace mémoire commun où toutes les autres applications ont accès (de manière locale ou distante). Une table de variables contient la liste des données présentes dans la mémoire et pour chacune d'elles, des informations sur son type, sa taille et sa date de modification (voir Figure 6.3). Une structure unique est associée à chaque type de variable de manière à lire de la même façon différentes données du même type.

Simulation : un simulateur permet de tester les algorithmes rapidement et sans avoir le matériel à portée de main. Ce simulateur a été conçu pour être le plus réaliste possible du point de vu des données capteurs et du système dynamique des différents robots. Ainsi, l'environnement a été modélisé en 3D avec le plus de détails possibles de telle sorte que nous pouvons appliquer des algorithmes de traitement d'images de la même manière que sur des images réelles.

Enregistrement et rejeu de données : afin de pouvoir analyser correctement les résultats d'une expérimentation il est indispensable de pouvoir enregistrer des données. Le fait de pouvoir rejouer ces données est aussi intéressant pour régler des paramètres et tester des applications sur des données réelles pré-enregistrées sans avoir à

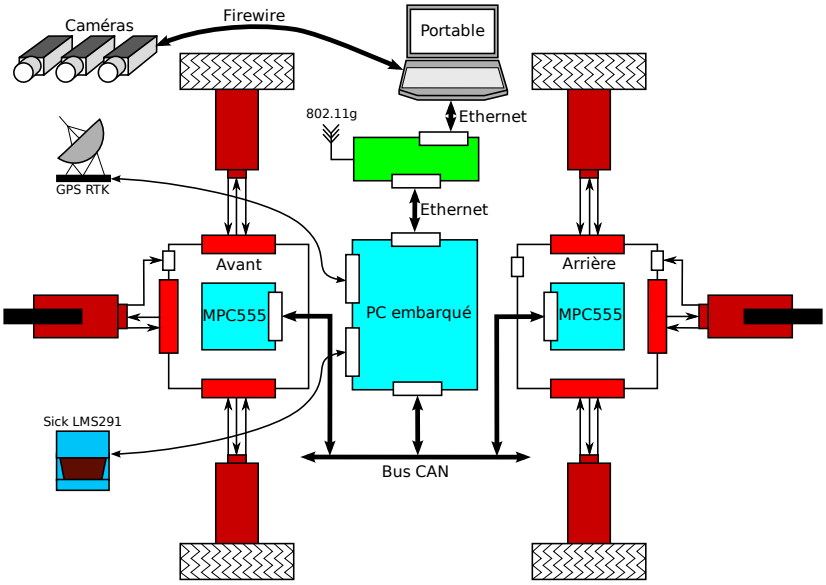


FIG. 6.2 – Architecture de la plateforme expérimentale *Cycab*.

relancer à chaque fois toutes les manipulations.

6.1.3 Plateforme de traitement et d'ordonancement des tâches

Pour effectuer des tâches de traitements d'images complexes comme l'ensemble du processus que nous avons proposé pour détecter et suivre des éléments dans une séquence vidéo, plusieurs remarques nous ont poussé à développer une nouvelle plateforme logicielle :

- Un processus de traitement d'image correspond à un enchaînement d'opérateurs plus ou moins complexes. La sortie d'un opérateur peut-être utilisé comme entrée d'un nouvel opérateur. Dans notre cas par exemple l'image est d'abord transformée en niveaux de gris, puis la pyramide gaussienne est calculée par filtrage successif de l'image précédente, cette pyramide est utilisée pour calculer la pyramide du laplacien, etc.
- La plupart des processus de traitement d'images utilisent des opérateurs de base, tel que le filtrage par convolution, le sous-échantillonnage, le seuillage, etc. Il est donc pratique d'avoir une collection d'opérateurs pouvant être connectés facilement les uns avec les autres.

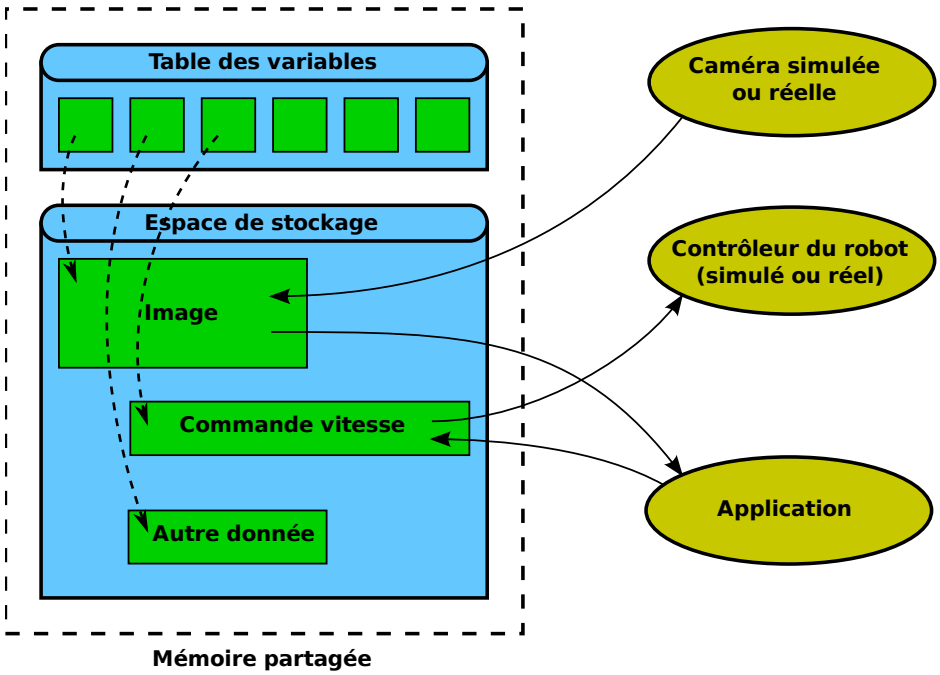
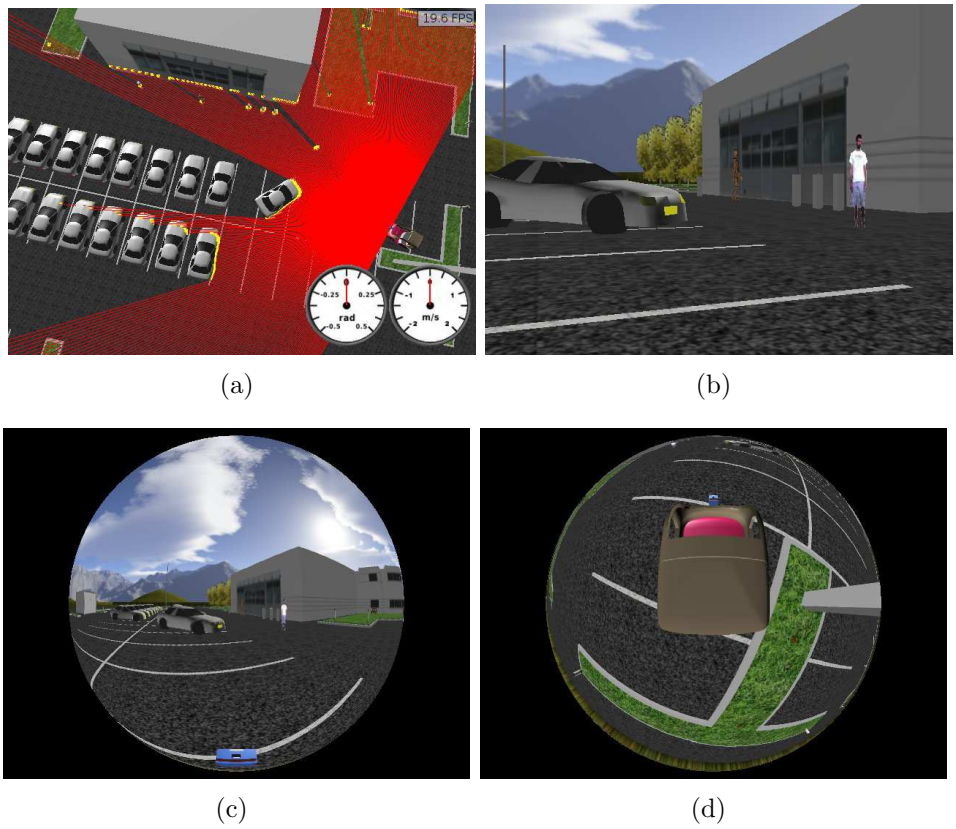


FIG. 6.3 – Architecture de la couche d'abstraction du matériel et de partage des variables.

- Pour optimiser le temps de calcul, il peut être intéressant de paralléliser certaines tâches indépendantes à un moment donné. Par exemple dans notre cas, les calculs de la pyramide du laplacien et du gradient peuvent être effectués en parallèle. Avec les machines modernes possédant souvent plusieurs processeurs, il est alors possible de gagner un facteur non négligeable sur le temps de traitement total.
- Pour tester et régler les algorithmes, il est utile de faire varier les différents paramètres et observer le résultat en temps réel. Les paramètres doivent donc être considérés comme des variables pouvant changer à chaque nouvelle étape de traitement.

D'après ces remarques, nous avons choisi de représenter les algorithmes de traitement d'images sous forme de graphes orientés. Les opérateurs sont représentés par des nœuds possédant des entrées et des sorties. Les arêtes permettent de relier les sorties des différents opérateurs vers les entrées des autres opérateurs. Chaque algorithme est alors décrit par un fichier de graphe qui peut être lu par un programme chargé d'ordonnancer et de paralléliser les différentes tâches. Chaque opérateur est implanté sous forme de module, ce qui permet une ré-utilisation très simple des différentes tâches.



(a) (b)

(c) (d)

FIG. 6.4 – Aperçu de l'environnement de simulation avec notamment un télémètre laser à balayage (a), une caméra perspective (b), une caméra grand angle (c) et une caméra omnidirectionelle (d).

Pour calculer les tâches pouvant êtres exécutées en parallèles, un niveau est associé à chaque tâche. On définit l'ensemble des prédécesseurs $Pred(t_i)$ de la tâche t_i comme l'ensemble des tâches dont une sortie au moins en relié à une entrée de t_i. Le niveau $n(t_i)$ associé à t_i est alors calculé de la manière suivante :

$$n_i = \begin{cases} 0 & \text{si } Pred(t_i) = \varnothing \\ \max\limits_{p \in Pred(t_i)} (n(p) + 1) & \text{sinon} \end{cases}$$

Ainsi, le niveau calculé définit l'ordre d'exécution des différentes tâches : si $n(t_i) < n(t_j)$ alors la tâche t_i doit être exécutée avant t_j, des tâches de niveaux identiques peuvent être exécutées en même temps et donc en parallèle. Toutefois, pour rendre le calcul possible, il faut imposer une contrainte sur le graphe en interdisant les cycles.

La Figure 6.5 montre un exemple d'application, chaque tâche est représentée par un bloc rectangulaire, les entrées sont affichées sur la partie supérieure et les sorties sur la partie inférieure du bloc. Dans cet exemple, le niveau de chaque tâche est croissant de haut en bas. Les tâches *LapPyramid* et *GradPyramid* peuvent être exécutées en parallèle.

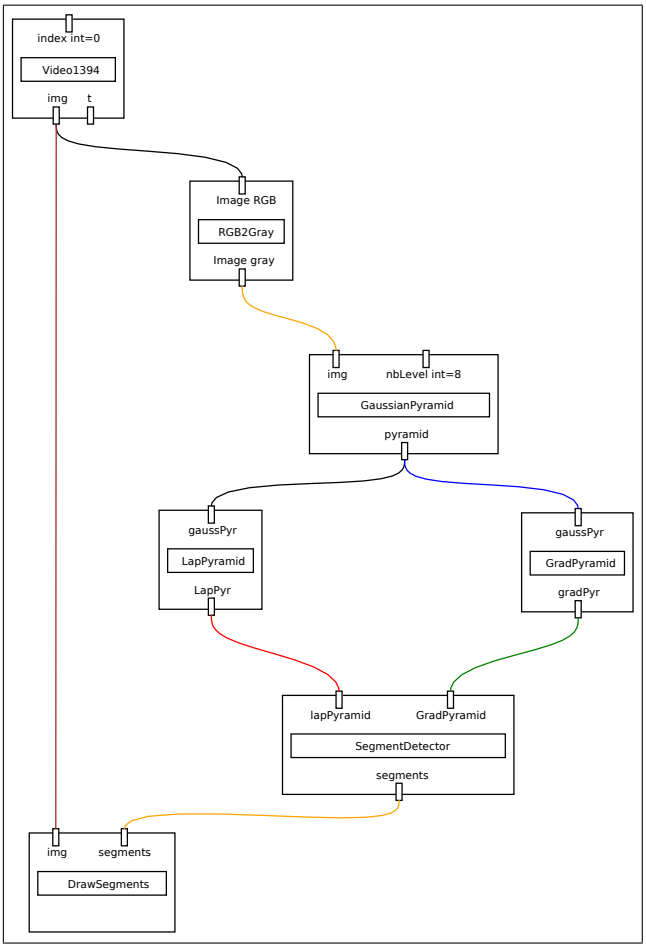


FIG. 6.5 – Exemple d'application de traitement d'image sous forme de graphe. Les blocs représentent les opérateurs ayant des entrées (en haut des blocs) et des sorties (en bas des blocs).

6.1.4 Utilisation du processeur graphique (GPU)

Pourquoi utiliser le GPU ?

Le processeur graphique est à la base un coprocesseur effectuant toutes sortes d'opérations destinées à afficher des objets en 2D ou 3D sur un écran.

Avec les besoins croissant en puissance de calcul de l'industrie graphique (jeux vidéos en particulier), ces processeurs ont rapidement évolué, l'objectif étant d'afficher toujours plus de polygones avec une meilleure résolution et avec plus d'effets de rendus (par exemple les éclairages complexes, les ombres, le rendu de flammes, etc). Ne pouvant pas augmenter indéfiniment la fréquences des processeurs, la solution choisie pour augmenter la puissance a été de paralléliser le plus possible les tâches. Cela a donc transformé l'architecture des processeurs qui sont alors dotés de plusieurs unités de calculs et de mémoires, ainsi que de contrôleurs chargés de gérer les tâches affectés à chaque unité. Pour améliorer les qualités de rendus et pour permettre de réaliser toute sorte d'effets, les processeurs graphiques deviennent alors programmables et les résultats des opérations peuvent être récupérés. Grâce à cette fonctionalité, le processeur graphique peut alors être utilisé pour des applications de natures différentes que le rendu graphique.

Aujourd'hui, les processeurs graphiques en terme de puissance brute sont plus puissants que les processeurs classiques (voir Figure 6.6), on peut donc comprendre facilement l'intérêt d'utiliser ces processeurs pour accélérer certains calculs. De plus, si on utilise ce processeur graphique, le processeur principal reste disponible pour effectuer d'autres traitements. Néanmoins, pour profiter de toute la puissance des processeurs graphique, les applications doivent être parallèles, c'est-à-dire que le même programme doit pouvoir être utilisé simultanément sur différentes données.

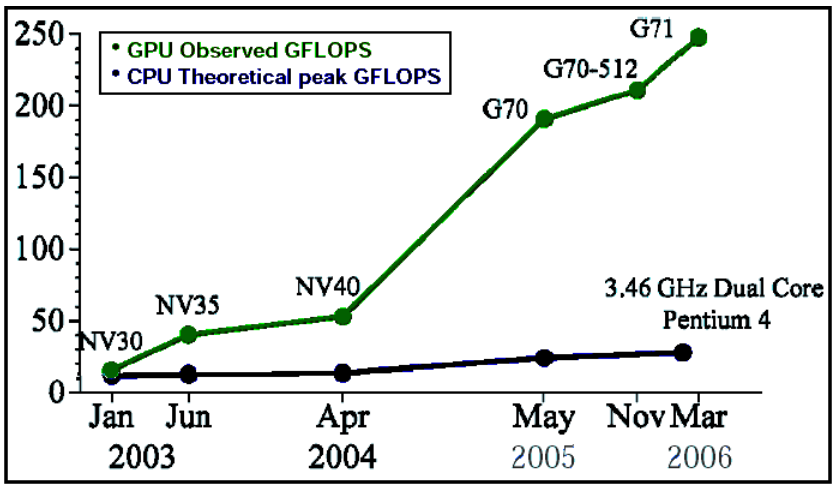


FIG. 6.6 – Evolution de la puissance théorique des processeurs graphiques nVidia par rapport aux processeur standard.

Pour le traitement d'image, les algorithmes classiques consistent à appliquer des traitements similaires sur chaque pixel de l'image, dans ce cas

il est donc possible de paralléliser les tâches en exécutant simultanément le même programme sur chaque pixel. Sur un processeur graphique qui compte de nombreuses unitées de calculs en parallèle, les calculs seront donc grandement accélérés.

Implémentation

L'architecture des processeurs graphiques ainsi que les algorithmes développés sont détaillés dans l'Annexe A. On peut toutefois faire remarquer que le parallélisme s'effectue principalement au niveau du pixel. La principale difficulté consiste à définir correctement les structures de données afin de pouvoir stocker et utiliser les variables sous forme de texture.

Temps de calculs

La Table 6.1 présente les temps de calculs des différentes étapes du processus de détection d'obstacles. On peut constater que la fréquence de traitement est de l'ordre de 15 images par seconde, ce qui correspond à l'ordre de grandeur des fréquences des caméras numériques. De plus,on peut remarquer que l'étape de détection des segments de crêtes est la plus coûteuse. Cette étape n'est nécessaire que pour l'initialisation des nouvelles cibles, il est possible d'effectuer cette étape de manière occasionnelle (1 ou 2 fois par secondes), ce qui permet de traiter plus de 20 images par seconde.

Etape	**Temps de calcul (ms)**
Lecture image caméra	2
Conversion texture OpenGL	1
Pyramide gaussienne	8
Pyramide laplacien	4
Pyramide gradient	4
Orientation des crêtes + filtrage fausse crêtes	2
Détection de segments de crête	22
Tracking des segments	12
Calcul du TTC	0.1
Total	62

TAB. 6.1 – Temps de calcul des différentes étapes sur GPU.

6.2 Résultats expérimentaux

6.2.1 Détection des segments de crête

Tout d'abord, pour évaluer le détecteur de segment de crête, nous allons étudier la répétabilité de notre détecteur sous différentes transformations, telles que la rotation, le changement d'échelle et la variation d'illumination.

La répétabilité est une notion introduite par Schmid, Mohr et Bauckage [SMB00] pour évaluer différents détecteurs de points d'intérêts. Pour mesurer la répétabilité pour une certaine transformation H, on applique le détecteur sur l'image I_1, on applique la transformation H à cette image et on applique à nouveau le détecteur sur l'image transformée I_2. La répétabilité correspond au pourcentage de points qui sont associés entre les deux images. De manière formelle, si $\tilde{x_1}$ et $\tilde{x_2}$ sont les ensembles de points détectés respectivement dans les images I_1 et I_2, on appelle $R_H(\epsilon)$ l'ensemble de points $x_1 \in \tilde{x_1}$ tel que $\exists\ x_2 \in \tilde{x_2}/dist(H(x_1), x_2) < \epsilon$ où $dist$ correspond à une certaine distance et ϵ est un nombre réel positif. La répétabilité du détecteur pour la transformation H et la précision ϵ est alors :

$$r_H(\epsilon) = \frac{|R_H(\epsilon)|}{|\tilde{x_1}|} \tag{6.1}$$

En fait, étant donné que tous les points transformés ne rentrent pas forcément dans l'image (par exemple avec un grossissement), il faut considérer uniquement les points situés dans la partie commune aux deux images.

Dans le cas de points d'intérêt, on peut utiliser toute sorte de distance (par exemple la distance euclidienne entre les points), dans le cas des segments de crêtes, la distance euclidienne ne signifie pas grand chose, nous avons donc utilisé la même mesure de similarité que celle définie en 4.2.5 pour détecter la fusion entre deux segments de crêtes. Les segments de crêtes sont alors caratérisés par leur centre 2D et une matrice de covariance, et deux segments sont associés si la divergence de Kullback-Leibler entre les deux distributions gaussiennes correspondantes est inférieure à ϵ.

Nous avons comparé les résultats avec deux détecteurs de points d'intérêt naturel de type Laplace et Harris multi-échelle. Le détecteur de Harris multi-échelle consiste à extraire les maxima locaux de l'opérateur de Harris dans l'espace d'échelle :

$$Harris(x, y, \sigma, s) = det\left(M(x, y, \sigma, s) - k \cdot trace(M(x, y, \sigma, s)^2)\right) \tag{6.2}$$

où M est la matrice suivante :

$$M(x, y, \sigma, s) = g_s * \begin{bmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{bmatrix} \tag{6.3}$$

où g_s est un noyau gaussien d'écart type s.

Ce détecteur est fondé sur le gradient, il extrait les points où la covariance du gradient est élevée, ce qui correspond bien à des coins.

Le détecteur de Laplace consiste à extraire les points dans l'espace d'échelle où le laplacien est maximum. Le laplacien correspondant à la somme des deux dérivées secondes en x et en y, ce détecteur extrait particulièrement les *blobs*, c'est-à-dire les tâches contrastées.

Le critère utilisé pour associer deux points extraits par ces deux détecteurs est le même que pour le détecteur de segments, mais la matrice de covariance est diagonale, ce qui revient à utiliser des cercles plutôt que des ellipses pour la distance de mahalanobis.

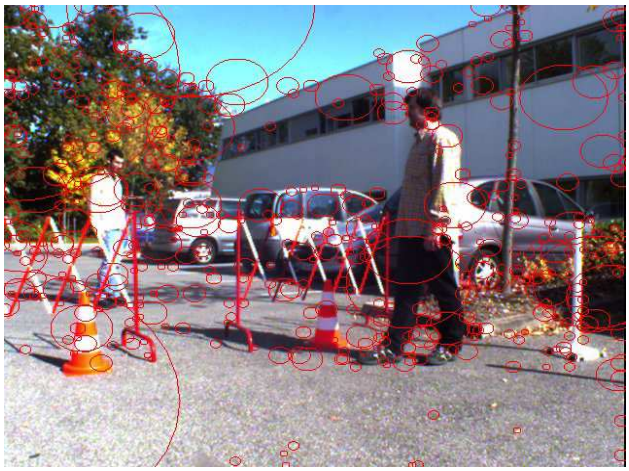
Protocole expérimental

Pour évaluer la répétabilité, nous avons utilisé une image de référence obtenue avec la caméra équipant le véhicule Cycab. La résolution de cette caméra est de 640x480 pixels. La scène capturée est une scène de type environnement urbain avec des piétons, des voitures, des poteaux, un bâtiment, ainsi que divers objets (voir Figure 6.7).

On applique ensuite les transformations à image de manière numérique (pour les transformations affines, on utilise un filtrage bilinéaire pour diminuer les effets de crénelage). On applique alors chaque détecteur sur l'image transformée et on calcule la répétabilité à l'aide de la Formule 6.1. Pour les expériences qui suivent, nous avons utilisé comme critère de précision $\epsilon = 1$.

FIG. 6.7 – Image de test utilisée.

(a) Détecteur de segments de crête.



(b) Détecteur de Laplace multi-échelle.

(c) Détecteur de Harris multi-échelle.

FIG. 6.8 – Résultats des différents détecteurs sur l'image de référence.

Rotation

Tout d'abord, nous allons étudier l'effet de la rotation sur les différents détecteurs. Les détecteurs de Harris et de Laplace sont réputés pour être robustes à la rotation. Le détecteur de segments de crête ne devrait pas être perturbé non plus par la rotation étant donné que la direction propre de chaque segment est calculée à l'aide de la Hessienne. Toutefois, on peut s'attendre à des problèmes de précision lors du calcul de cette direction propre dus à des problèmes de discrétisation et d'aliasing.

Comme on peut le voir sur la Figure 6.9, les trois détecteurs donnent des résultats similaires, le détecteur de Harris étant légèrement plus performant que les autres détecteurs et le détecteur de Laplace légèrement derrière.

Changement d'échelle

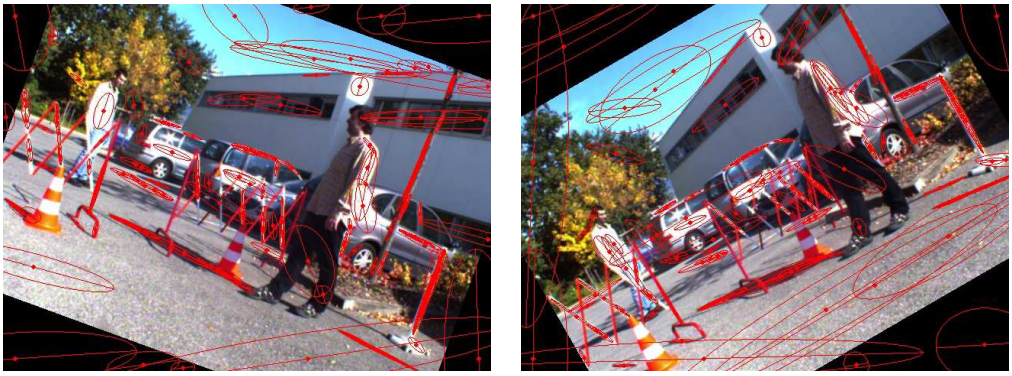
Pour étudier l'effet du changement d'échelle, nous effectuons un grossissement de l'image avec plusieurs facteurs d'échelles. Pour couvrir toute l'image, nous effectuons aussi plusieurs translations lorsque le rapport d'échelle est supérieur à 1. On peut voir sur la Figure 6.10 la répétabilité en fonction du facteur d'échelle.

Pour des facteurs d'échelle inférieurs à 1, seul le détecteur de Harris donne de bons résultats, les détecteurs de segments et de Laplace chutent car les éléments détectés à une faible échelle ne peuvent plus être détectés.

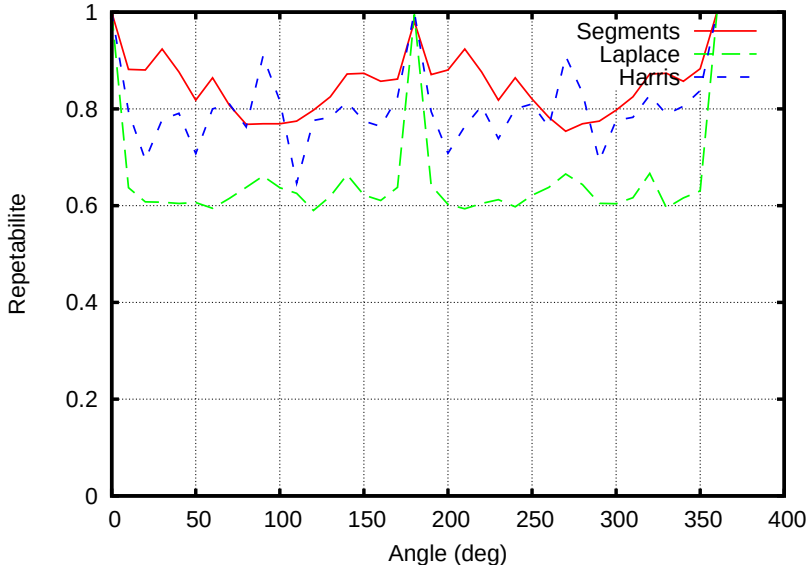
Pour des facteurs d'échelle supérieur à 1, le détecteur de segments de crête est plus performant que les deux autres détecteurs. Le détecteur de Harris a une répétabilité assez élevée $(0,8)$ jusqu'à un facteur d'échelle de 2 puis chute à $0,6$, cela provient du fait que l'échelle de détection n'est pas vraiment liée à l'échelle des points et on ne peut donc pas considérer cette échelle comme une taille caractéristique. Le détecteur de Laplace obtient des scores moyens $(0,7)$ mais ces scores varient très peu avec la variation d'échelle, cela signifie que l'on a une erreur constante. Cela provient des points instables qui sont situés sur des segments de crête, la moindre variation de l'image causant une perte de la détection (on peut aussi observer cela sur la courbe de répétabilité en fonction de la rotation). Cependant, la répétabilité variant très peu lors de grands rapports d'échelle, on peut considérer que le détecteur de Laplace est robuste à ces transformations.

Variation d'illumination

Comme on peut le voir sur la Figure 6.11, les trois détecteurs sont fortement liés à l'intensité lumineuse. Lorsque l'on multiplie l'intensité de chaque pixel par un facteur inférieur à 1, on diminue le contraste ce qui fait



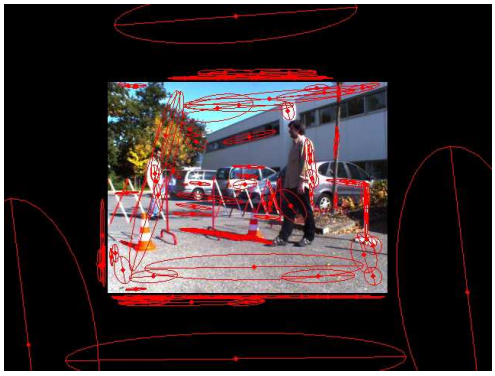
(a) (b)



(c)

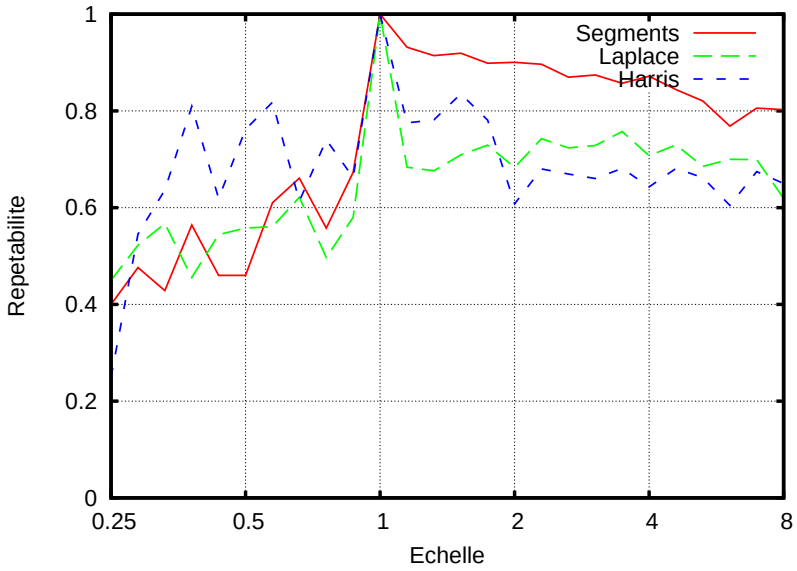
FIG. 6.9 – Exemple de détection de segment de crêtes avec une rotation de 20 degrés (a) et de -50 degrés (b). La Figure (c) retrace la répétabilité en fonction de l'angle de rotation pour les trois détecteurs (segments de crête, Harris et Laplace multi-échelle). On constate que les trois détecteurs donnent des résultats similaires, avec néammoins un désavantage pour le détecteur de Laplace.

que les dérivées premières et secondes qui sont utilisées par les détecteurs sont aussi multipliées par le facteur d'intensité, et tous les détecteurs perdent donc en répétabilité. Lorsqu'on multiplie l'intensité par un facteur supérieur à 1, les valeurs d'intensité saturent sur leur valeur maximale et on perd ainsi des



(a)

(b)



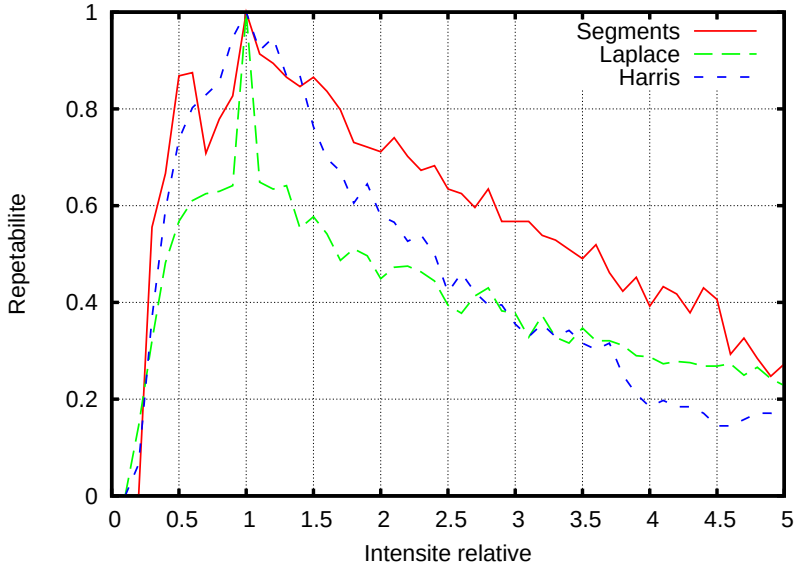
(c)

FIG. 6.10 – Exemple de détection de segment de crêtes avec un facteur d'échelle de $0,57$ (a) et de $1,74$ (b). La Figure (c) retrace la répétabilité en fonction de l'échelle pour les trois détecteurs (segments de crête, Harris et Laplace multi-échelle). On constate que le détecteur de segments donne de meilleurs résultats que les deux autres détecteurs.

points ou des crêtes sur les zones lumineuses.

(a)

(b)



(c)

FIG. 6.11 – Exemple de détection de segment de crêtes avec un facteur d'intensité de $0,5$ (a) et 2 (b). La Figure (c) retrace la répétabilité en fonction du facteur d'intensité pour les trois détecteurs (segments de crête, Harris et Laplace multi-échelle). Etant donné que les trois détecteurs détectent les zones où le contraste varient fortement, une variation d'intensité (par un facteur multiplicatif) fait varier le contraste et donc augmente ou diminue le nombre de points détectés.

6.2.2 Suivi des segments de crête

Evaluation de l'erreur statique

Tout d'abord, pour évaluer la précision et la stabilité de l'algorithme de suivi de segments de crête, nous avons exécuté cet algorithme sur une image fixe (en reprenant l'image de test utilisée pour évaluer la détection de segments de crête). Nous avons alors évalué l'écart type des différentes paramètres des segments de crête suivis : position spatiale et en échelle du centre, composantes du demi-segment. Il est évident que le bruit introduit lors de l'application du modèle de transition a une conséquence sur les variations observées, nous avons donc fait des mesures pour différentes valeurs de bruit. Les résultats obtenus sont affichés dans le Tableau 6.2.

Nous pouvons remarquer plusieurs points intéressants sur ces résultats :

- Le bruit sur la position spatiale du centre des segments se répercute sur la position estimée du centre, mais joue aussi significativement sur l'échelle estimée, ce qui signifie qu'une imprécision spatiale se répercute sur l'échelle. Cela s'explique par le fait que pour un objet uniforme dans l'image, son échelle caractéristique n'est pas constante sur toute sa surface (l'échelle est plus grande au milieu que sur les bords).
- De même les variations en échelle influent sur la précision spatiale, cela s'explique de la même manière que la remarque précédente.
- Les variations d'angle perturbent tous les points des segments de crête, ces variations influent donc sur tous les paramètres. Dans le cas d'une scène filmée par un véhicule de type voiture, les variations d'angles des éléments observés sont relativement faibles, il sera donc préférable de conserver un bruit très faible pour la variation d'angle.
- Seul le bruit appliqué directement sur les paramètres du demi-segments (angle et longueur) a une réelle influence sur le demi-segment.

Evaluation sur une scène dynamique

Pour tester notre algorithme sur une scène dynamique, nous avons enregistré une séquence video avec une caméra fixée sur le robot mobile Cycab. La scène comporte un certain nombre de piétons et de voitures qui sont quasiment immobiles, toutefois, le robot effectuant une trajectoire, toute la scène est mobile dans l'espace image. La séquence vidéo comporte 477 images à environ 20 images par seconde (voir Figure 6.12).

La Figure 6.13 illustre le résultat de la phase de suivi. Pour obtenir ce résultat, nous avons limité le nombre d'objets suivis à 32 (principalement pour des raisons de lisibilité). Les objets suivis sont représentés par des ellipses rouges avec l'axe principal en bleu. Le numéro affiché en jaune

Ecart type du bruit introduit dans le modèle de transition				Ecart type observé		
position du centre (pixel)	$1/\sigma$	angle (radian)	variation de taille ($s^{\Delta t}$)	centre (pixel)	échelle (pixel)	demi-segment (pixel)
(5 ; 5)	0,001	0,01	0,1	(4,85 ; 3,52)	2,18	(2,17 ; 1,23)
(2 ; 2)	0,001	0,01	0,1	(2,53 ; 2,35)	1,13	(1,97 ; 1,42)
(1 ; 1)	0,001	0,01	0,1	(1,53 ; 1,75)	0,99	(1,75 ; 1,33)
(2 ; 2)	0,01	0,01	0,1	(3,27 ; 2,56)	2,12	(2,94 ; 2,12)
(2 ; 2)	0,005	0,01	0,1	(2,54 ; 2,41)	1,29	(1,46 ; 1,42)
(2 ; 2)	0,0001	0,01	0,1	(0,99 ; 1,30)	0,83	(1,5 ; 1,78)
(2 ; 2)	0,001	0,001	0,1	(2,10 ; 1,80)	0,92	(1,41 ;1,07)
(2 ; 2)	0,001	0,1	0,1	(2,96 ; 3,10)	1,73	(3,04 ; 2,36)
(2 ; 2)	0,001	0,5	0,1	(3,25 ; 3,37)	2,34	(5,41 ; 3,69)
(2 ; 2)	0,001	0,01	0,5	(2,23 ; 2,28)	1,39	(4,48 ; 5,08)
(2 ; 2)	0,001	0,01	0,01	(2,57 ; 2,30)	0,92	(1,32 ; 1,17)

TAB. 6.2 – Ecarts types des paramètres des segments de crête suivis en fonction des paramètres du modèle de transition. La scène observée est statique et les résultats sont obtenus en moyennant les écarts types sur l'ensemble des 48 segments suivis. Chaque filtre compte 1024 particules.

correspond à l'identifiant de l'objet. De plus, seul les objets suivis depuis plus de trois images sont affichés.

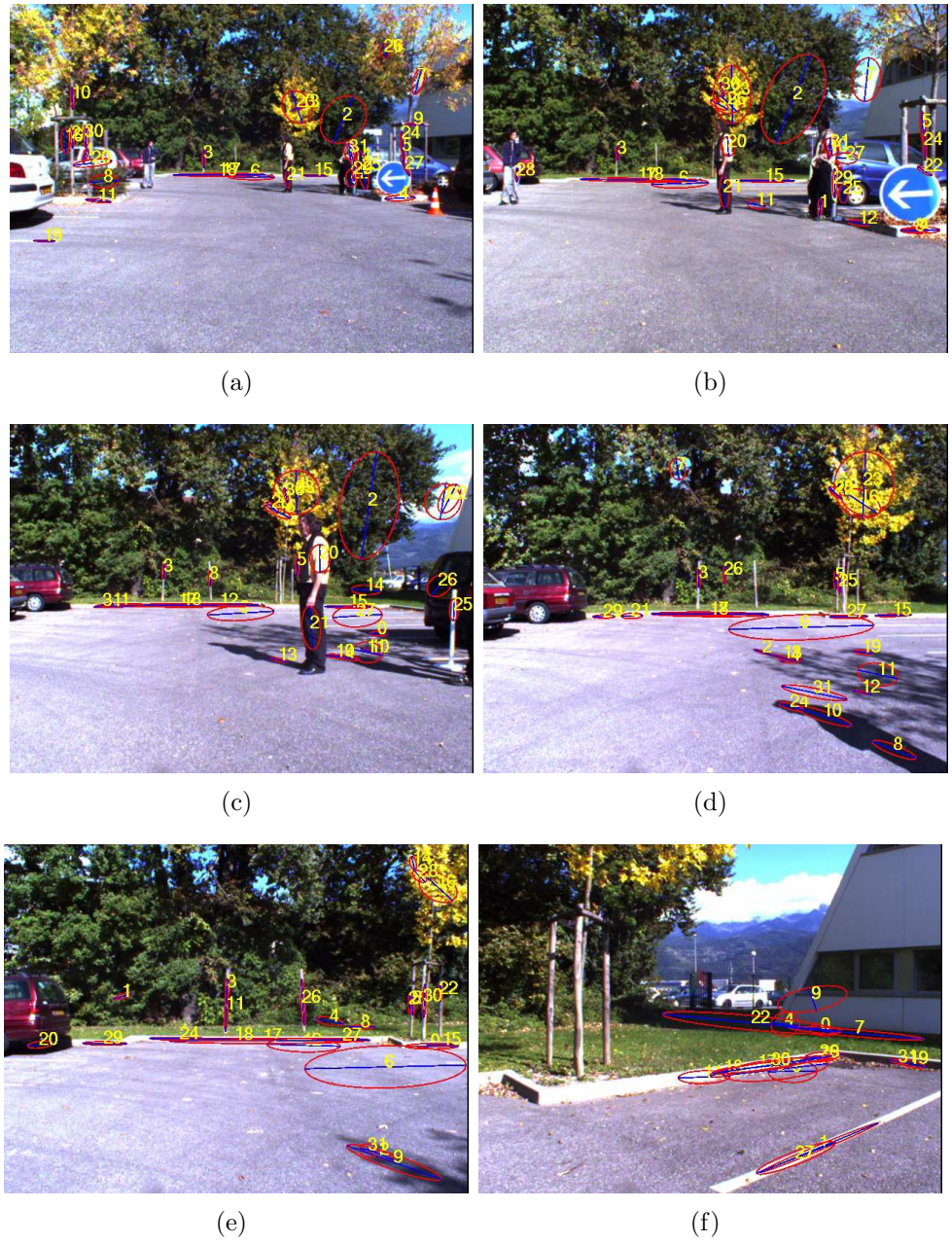
On peut constater que les deux piétons de droite sont bien détectés (identifiants 21, 20, 1 et 31) et donnent lieu à des cibles très stables (l'objet 21 est suivi du début jusqu'au dépassement du piéton), les principaux éléments suivis correspondant au buste et aux jambes des piétons. Par contre le piéton de gauche n'est pas détecté, ce qui s'explique par le très faible contraste par rapport au reste de l'image, en effet le piéton a les jambes d'une couleur très proche de celle de la route et le buste foncé comme les arbres en arrière plan.

On peut noter aussi la présence de certaines cibles suivies ne correspondant à aucun objet physique, comme la cible 6 (au centre à droite dans les images (d) et (e)). Ces cibles sont généralement observées dans les grandes échelles et correspondent en fait à des artefacts provoqués par d'autres objets. La cible sur la route peut en effet s'expliquer par la présence de l'ombre et de la zone sombre au dessus, ce qui fait ressortir la zone claire de la route.

FIG. 6.12 – Séquence d'images : la caméra se déplace sur un parking en présence de trois piétons.

Ensuite, nous nous sommes aperçus que les cibles présentes dans les petites échelles (correspondant à des objets fins) sont moins stables que dans les grandes échelles. Cela s'explique naturellement en considérant la fonction d'observation utilisée par les filtres à particules : le laplacien de l'image a en effet beaucoup plus d'extrema locaux dans les petites échelles que dans les grandes échelles. Lorsque la caméra bouge, le filtre à particules a donc plus de mal à converger vers le bon extremum, ce qui provoque des divergences des particules.

Enfin, vers la fin de la séquence, le véhicule tourne fortement vers la droite, ce qui provoque un déplacement important des pixels vers la gauche, et on constate que la majorité des segments de crêtes verticaux ont été perdus (comme par exemple les segments de crêtes présents sur le tronc et les piquets de l'arbre). Les cibles horizontales sont mieux conservées car une bonne portion des segments de crête reste superposée à la crête lors d'un mouvement horizontal et la fonction d'observation reste donc élevée. Pour améliorer les performances du suivi, il serait possible de tenir compte du déplacement et plus précisément de la rotation du robot (mesurée par



(a) (b)

(c) (d)

(e) (f)

FIG. 6.13 – Résultat de l'algorithme de suivi.

exemple avec un gyroscope) pour pouvoir prédire, au moins partiellement, le mouvement des points dans l'image. Cependant cette méthode nécessite une bonne synchronisation entre les capteurs et les acquisitions caméra, et requiert en plus un calibrage de la caméra. Une seconde possibilité plus simple

est de tenir compte de la cinématique du véhicule et du fait que celui-ci se déplace sur un sol plat, ce qui fait que les pixels se déplacent beaucoup plus horizontalement que verticalement. On peut donc modifier uniquement les paramètres du modèle de transition et insérer un bruit plus important sur le déplacement horizontal.

Pour connaître plus précisément les principales causes de perte des segments de crêtes, nous avons compté le nombre de filtres réinitialisés et la cause de cette réinitialisation. Nous rappelons qu'il y a trois causes possibles : la sortie de l'image (ou de l'espace d'échelle calculé), la fusion avec une autre cible (distance trop faible entre deux cibles suivies) et lorsque la probabilité d'observation et inférieure à un seuil. Le résultat est affiché sur la Figure 6.14. On remarque que le nombre de fusions est plus élevé que les autres cas de perte. En regardant plus en détail la séquence on s'aperçoit que ces fusions sont dues au fait que plusieurs cibles différentes sont fréquemment initialisées sur la même crête mais à des extrémités différentes. La fonction d'évaluation a tendance à faire converger les deux cibles au centre de la crête et on détecte alors une fusion. Ensuite, le nombre de pertes dues à une chute de la probabilité d'observation reste faible au début de la séquence et augmente fortement vers la fin de la séquence. Cela vient du fait qu'à la fin de la séquence, le véhicule tourne et le déplacement des points dans l'image augmente alors subitement. Le modèle dynamique des filtres n'est donc plus assez réactif pour suivre les cibles et celles-ci sont alors perdues.

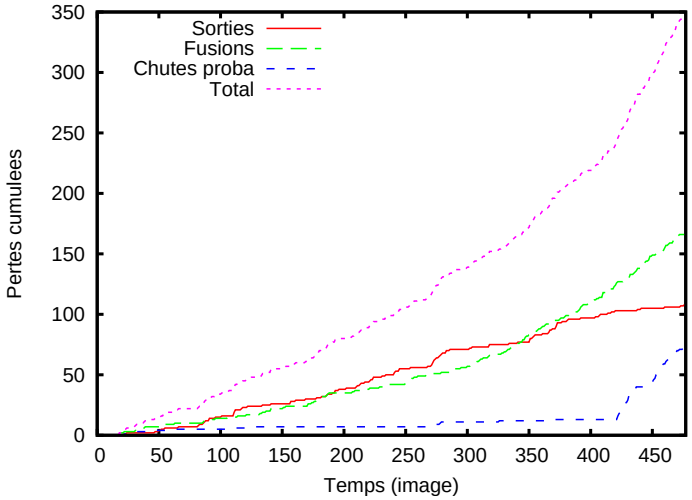


FIG. 6.14 – Evolution du nombre de segments perdus au cours de la séquence. Les différentes courbes représentent les différents cas de pertes (sortie de l'image, fusion avec une autre cible et chute de la probabilité d'observation).

6.2.3 Evaluation de la taille estimée

Nous avons montré dans le chapitre 5 que le temps avant collision peut être calculé directement à partir du changement d'échelle à l'aide de la formule :

$$\tau = \frac{\sigma}{\frac{d_\sigma}{d_t}} \tag{6.4}$$

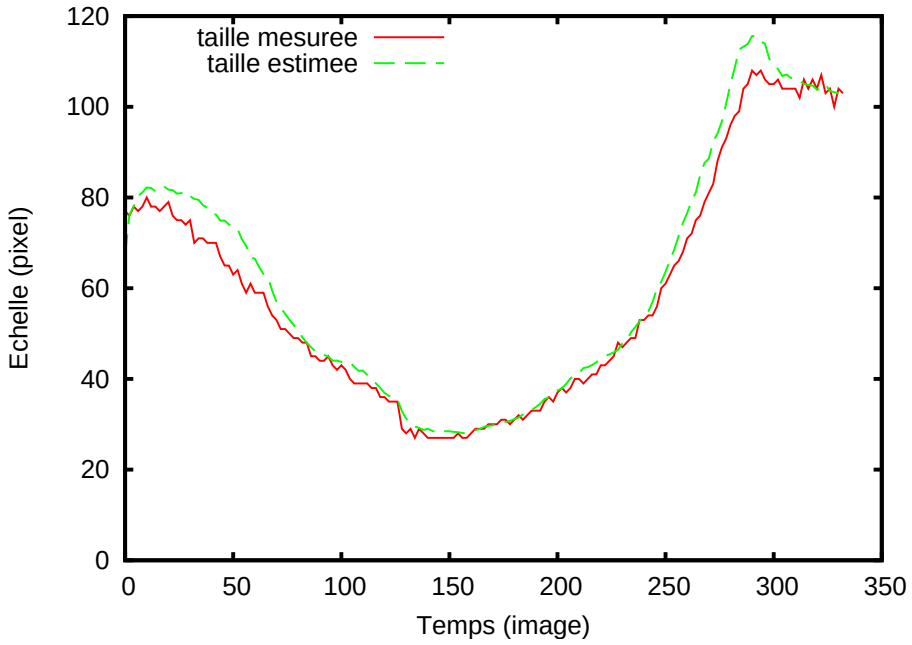
Pour évaluer correctement le temps avant collision, il est donc impératif d'estimer précisément l'échelle (ou la taille) des objets suivis. Nous allons donc tout d'abord évaluer la précision en échelle de notre algorithme de suivi des segments de crête.

Pour cela, nous allons filmer un motif très simple constitué d'un rectangle noir sur fond blanc en faisant varier la distance entre la caméra et le motif. Nous initialiserons un tracker de segment au centre du motif, puis, pour chaque image nous comparerons la taille estimée par le tracker et la largeur du rectangle dans l'image mesurée à la main. L'échelle du segment suivi sera extraite en calculant l'échelle moyenne de toutes les particules pondérée par leur poids. La fonction d'observation du filtre à particule est fondée sur le laplacien normalisé, ce qui signifie que le poids des particules sera maximal aux échelles qui maximisent le laplacien normalisé, et donc à l'échelle caractéristique du point considéré. Au centre du segment, la taille caratéristique doit correspondre à la largeur du segment et on devrait donc avoir des mesures très proches.

La Figure 6.15 montre le résultat des échelles mesurées à la main et celles obtenues avec le tracker. Comme prédit, les deux courbes sont très proches, ce qui signifie que le tracker calcule bien l'échelle caractéristique. On peut aussi noter que la courbe obtenue par le tracker est plus lisse que celle mesurée à la main, ce qui montre que le tracker filtre bien les résultats grâce au modèle dynamique. Enfin, on constate que les échelles renvoyées par le tracker sont légèrement plus grandes que celles mesurées à la main, surtout lorsque l'échelle varie fortement. Cela peut s'expliquer de deux manières :

- L'amplitude du laplacien normalisé en fonction de l'échelle n'est pas symétrique (voir Figure 3.4), la laplacien normalisé décroît moins vite vers les grandes échelles, ce qui peut biaiser et augmenter la valeur de la moyenne par rapport à l'échelle caractéristique réelle.
- Ensuite, lorsque le motif se déplace dans l'image, l'apparence locale de l'image varie plus dans les petites échelles que dans les grandes échelles, les particules présentes dans les grandes échelles vont donc être plus stables que dans les petites échelles, ce qui induit un second biais faisant augmenter l'échelle moyenne des particules.

(a) Images extraites de la séquence vidéo.



(b) Echelles obtenues par le tracker de segment et en mesurant à la main pour chaque image de la séquence.

FIG. 6.15 – Comparaison entre la taille estimée par notre algorithme de suivi et la taille réelle (mesurée à la main) d'un segment de crête.

6.3 Utilisation pour la commande de véhicules

Dans la partie précédante, nous avons évalué et validé les phases de détection et de suivi des structures. Nous allons maintenant utiliser l'algorithme d'estimation du temps avant collision dans deux applications de navigation visuelle. La première application utilisera une seule cible, le but est de démontrer qu'il est possible de stopper le véhicule avant d'entrer en contact avec cette cible. Dans la seconde application, nous allons réaliser

une tâche de navigation réactive en utilisant l'ensemble des cibles suivies et en estimant le TTC de chaque cible pour calculer de manière réactive la trajectoire du robot.

6.3.1 Arrêt du véhicule avant collision

Dans la première application, l'objectif est de prouver qu'il est possible de s'arrêter avant une collision avec un obstacle prédéfini en utilisant le TTC calculé à partir de l'échelle dans l'espace image. Pour cela, nous allons placer une caméra embarquée sur un véhicule. Un obstacle sera placé devant le véhicule et sa position initiale dans l'image sera indiquée à la main. Nous allons utiliser deux séquences vidéos dans lesquelles l'obstacle correspond à un piéton fixe situé face à la caméra. Le véhicule se rapproche du piéton à vitesse constante, ce qui implique que le temps avant collision du piéton diminue de manière linéaire avec le temps.

La première séquence est une séquence simulée, ce qui permet de fixer de manière très précise la vitesse de la caméra (1m/s) et d'avoir une vérité terrain du temps avant collision. La seconde séquence a été obtenue en plaçant la caméra sur le véhicule Cycab et en donnant une vitesse de consigne constante de 1 m/s au contrôleur du véhicule. La position du piéton par rapport à la caméra n'est pas connue de manière précise à tout instant, mais la vitesse du robot étant constante, le temps avant collision doit varier linéairement.

Afin d'avoir un modèle de comparaison, nous avons utilisé une seconde méthode pour calculer le changement d'échelle, et donc le temps avant collision. Cette méthode consiste à calculer pour chaque image, la transformation affine entre l'objet observé dans l'image courante et l'objet observé sur un image de référence. On peut alors extraire le changement d'échelle de la transformation affine et obtenir ainsi l'échelle de l'objet dans l'image courante (relativement à la référence). La transformation affine est calculée en détectant les points d'intérêts SIFT sur l'image de référence et l'image courante puis en faisant de la mise en correspondance (grâce au descripteur SIFT) entre les deux ensembles de points. Pour le calcul de la transformation affine à partir de ces points de contrôles, on se référera à [AAT07, AT08]. Cette méthode est utilisée ici car elle représente bien les méthodes classiques utilisant un ensemble de points d'intérêts SIFT bien à la mode depuis quelques années.

La Figure 6.16 présente les résultats obtenus avec les données simulées. Pour calculer le TTC avec le suivi de segments de crête, nous initialisons un filtre au début de la séquence sur le buste du piéton. Pour utiliser la seconde méthode, nous avons tout d'abord détecté les points de contrôles SIFT sur

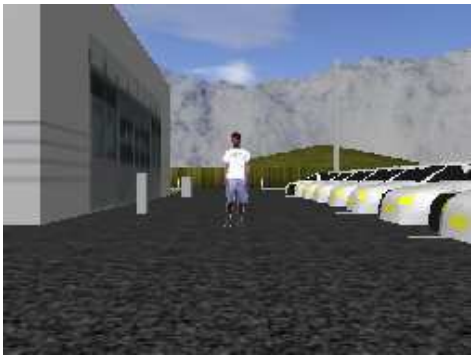
une image du piéton, puis pour chaque image de la séquence, on calcule le changement d'échelle par rapport à l'image de référence. On remarque que les résultats obtenus en utilisant notre algorithme de suivi de crêtes sont assez bruités lorsque l'obstacle est éloigné (en début de séquence), mais lorsque l'obstacle est proche, le TTC calculé est très proche du TTC réel. La seconde méthode donne des résultats très proches de la réalité, même sans filtrage (paramètre d'oublis faible), toutefois, la transformation affine ne peut pas être calculée lorsque l'objet est trop loin car on ne détecte pas assez de points SIFT sur l'obstacle. Le temps avant collision ne peut donc être calculé que lorsque l'obstacle est suffisamment large dans l'image.

La Figure 6.17 présente les résultats obtenus en calculant le TTC dans la scène réelle. Dans le début de la séquence, la caméra est fixe, ce qui se traduit par une échelle constante, le TTC calculé est donc très instable et saute entre des valeurs négatives et positives élevées. Dès que la caméra commence à avancer, l'échelle augmente de telle sorte que son inverse varie linérairement, ce qui traduit bien un vitesse constante et le TTC calculé par notre méthode devient stable et converge bien vers le TTC réel. En utilisant les points SIFT, il est impossible de calculer la transformation affine avant que l'obstacle soit assez proche à cause du manque d'associations possible. Cependant, lorsque le piéton est assez proche, le TTC calculé est bien estimé.

A partir de ces deux expériences, nous pouvons voir qu'en fixant un seuil sur le TTC on peut stopper le véhicule avant d'entrer en collision. Par exemple, si on fixe le seuil à 1,5 secondes (30 images dans les expériences), on va bien s'arrêter avant de toucher l'obstacle, ce qui permet de valider notre approche.

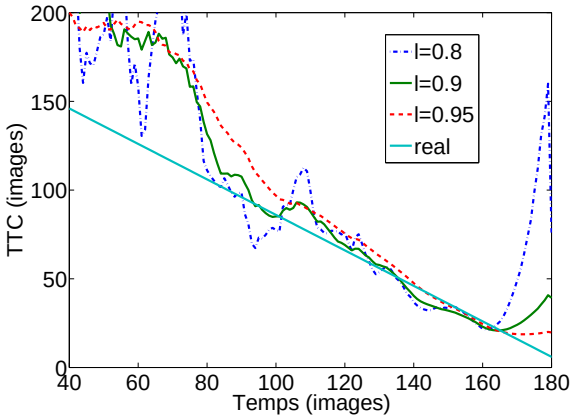
Pour discuter des points forts et des faiblesses de notre méthode par rapport à la méthode de comparaison utilisant des points d'intérêt, on peut faire ressortir plusieurs éléments. Tout d'abord, notre méthode est mieux adaptée aux objets peu texturés. En effet, si l'objet est bien texturé, on va détecter plusieurs segments de crêtes dans les petites échelles qui sont difficiles à suivre. Par contre, les méthodes utilisant les points d'intérêts SIFT vont pouvoir utiliser de nombreux points de contôle et donc permettre une bonne estimation du TTC. Par contre, si l'objet est peu texturé, une seule crête est détectée, le suivi de crête est donc aisé et l'échelle caractéristique est bien estimée. Avec la seconde méthode, on détecte peu de points d'intérêt et le TTC est donc impossible à estimer. Ensuite, notre détecteur ayant des bonnes propriétés d'invariances aux variations d'échelles, les éléments peuvent être suivis même lorsque les objets varient fortement en échelle. Pour la seconde

méthode, si l'échelle change fortement, l'apparence locale des points d'intérêts peut changer considérablement et la mise en correspondance des points peut échouer, le TTC peut donc s'avérer impossible à calculer dans ce cas.

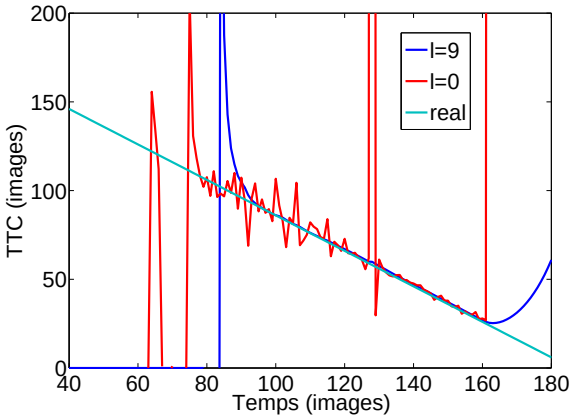


(a) Première image

(b) Dernière image

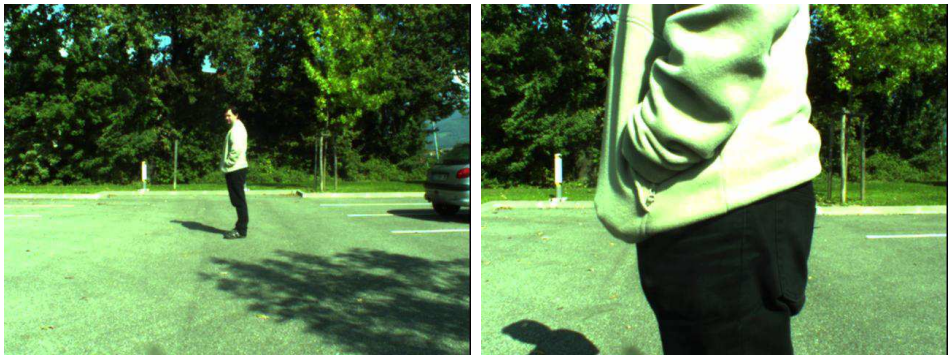


(c) TTC, par suivi de crête.

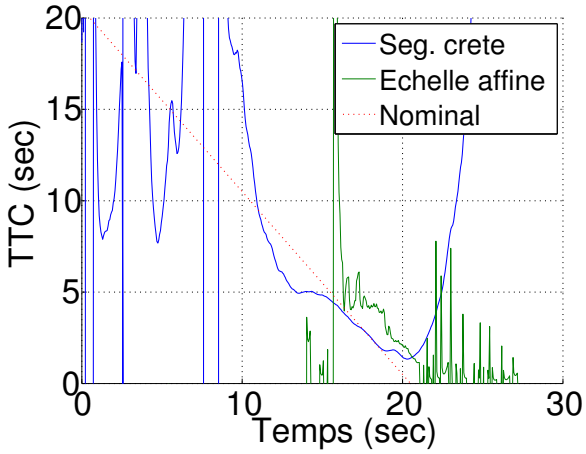


(d) TTC, par calcul de la transformation affine des points SIFT.

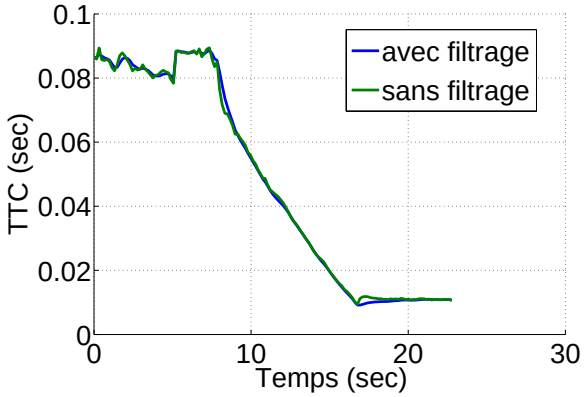
FIG. 6.16 – (a) première image et (b) dernière image de la séquence simulée composée de 140 images avec un déplacement de 8,26m à 1,21m. (c) (d) TTC calculé avec les deux algorithmes et en utilisant le filtre RLS avec différentes valeurs d'oublis.



(a) Première image (b) Dernière image



(c) Temps avant collision.



(d) Inverse de l'échelle du segment de crête suivi.

FIG. 6.17 – (a) première image et (b) dernière image de la séquence réelle composée de 140 images avec un déplacement de 6m50 à 1m50 du piéton. (c) TTC calculé avec les deux algorithmes. (d) Inverse de l'échelle.

6.3.2 Navigation réactive

Cette seconde application consiste à contrôler de manière réactive l'angle de braquage d'un robot mobile qui avance à vitesse constante, de manière à éviter les obstacles.

Pour effectuer cette application, nous avons utilisé une approche bayésienne d'évitement d'obstacle décrite dans [KPBM03]. Cette méthode consiste à calculer une fonction de probabilité de la commande du robot en fonction des observations et de la commande désirée (pour suivre une trajectoire par exemple). Dans notre cas, la commande du robot correspond à l'angle de braquage et sera représentée par la variable Φ. Les observations correspondent aux cibles suivies. Pour restreindre le nombre d'observations et conserver uniquement les données pertinentes, nous ne conservons que les cibles ayant un TTC positif et inférieur à 10 secondes. Une observation i, parmi les k observations conservées, sera alors caractérisée par sa position horizontale dans l'image X_i comprise entre -1 et 1 (0 correspondant au centre de l'image) et son TTC T_i. La commande désirée correspondant à un angle de braquage, sera représentée par la variable Φ_d.

Pour éviter les obstacles, nous avons suivi une approche proscriptive, comme recommandé dans [KPBM03]. Il s'agit alors de définir des distributions de probabilité qui "interdisent" certaines consignes qui nous amèneraient vers les obstacles. Pour un obstacle i, on définit une variable de diagnostic I_i qui exprime la compatibilité en terme de sécurité entre Φ et (X_i, T_i). Il s'agit alors de construire la distribution $P(I_i = 1|\Phi X_i T_i)$. Nous allons construire cette fonction en utilisant une gaussienne inversée, centrée proportionellement à la position horizontale de la cible et avec une covariance inversement proportionelle au TTC. On peut justifier ces choix de la manière suivante :

- Le choix de la gaussienne inversée s'explique par le choix d'un modèle proscriptif, on laisse à 1 toutes les commandes faisables mais il faut interdire les commandes dangereuses.
- La moyenne est choisie de telle sorte qu'un obstacle à droite de l'image empêche le robot de tourner de ce coté. Cependant, un obstacle au centre ou peu excentré doit laisser la possibilité de tourner des deux cotés.
- Plus le TTC est faible plus l'obstacle est proche et dangereux, il faut donc augmenter la largeur ce qui augmente le nombre de commandes dangereuses.

La distribution s'exprime donc de la manière suivante :

$$P(I_i = 1|\Phi X_i \tau_i) = 1 - \alpha e^{-\frac{1}{2}(\Phi - \alpha_x X_i)^2 (\alpha_t T_i)^2} \tag{6.5}$$

où α, α_x et α_t sont trois paramètres à définir. α représente la confiance associée à la détection et α_x et α_t sont des coefficients de proportionalités dépendant de la caméra et de la vitesse du robot.

Pour rester en accord avec la commande désirée, on ajoute une nouvelle variable de diagnostic I_d pour exprimer la validité de la commande par rapport à la commande désirée. On définit alors la distribution $P(I_d = 1|\Phi\Phi_d)$ sous forme de cloche évasée :

$$P(I_d = 1|\Phi\Phi_d) = e^{-\frac{1}{2}\left(\frac{\Phi-\Phi_d}{\sigma_\Phi}\right)^2} \tag{6.6}$$

La valeur de σ_Φ représente l'importance à accorder à la commande, plus σ_Φ est petit, plus on va essayer de suivre la consigne et moins on va avoir tendance à éviter les obstacles.

Pour calculer la commande, il faut alors fusionner les modèles d'évitement et de suivi de consigne. Pour cela, on construit la distribution de probabilité $P(\Phi|\Phi_d X_1..X_k T_1..T_k[I_i = 1]_{i=1..k}[I_d = 1])$, calculée de la manière suivante :

$$P(\Phi|\Phi_d X_1..X_k T_1..T_k[I_i = 1]_{i=1..k}[I_d = 1]) \propto P([I_d = 1]|\Phi\Phi_d) \prod_{i=1..k} P(I_i = 1|\Phi X_i T_i) \tag{6.7}$$

A chaque nouvelle image, on va alors construire cette distribution et choisir la commande Φ qui maximise cette distribution. Un exemple de distribution obtenue est illustré par la figure 6.18. On peut noter qu'avec 2 obstacles sur la droite et un obstacle sur la gauche, le robot va tourner légèrement à gauche pour éviter l'obstacle ayant le plus petit TTC, mais sans braquer trop pour éviter également l'obstacle de gauche.

Nous avons testé cette application sur des données simulées. Le véhicule avec une caméra embarquée est placé dans un environnement virtuel composé du sol et plusieurs obstacles fixes, dont un bâtiment et quatre piétons (voir Figure 6.20). Le véhicule se déplace à 0,5 m/s et l'angle de braquage est borné entre -12 et 12 degrés. Comme on peut le voir sur la figure, lors des différents tests effectués, le véhicule a suivi plusieurs trajectoires différentes mais le véhicule n'est entré en contact avec aucun obstacle.

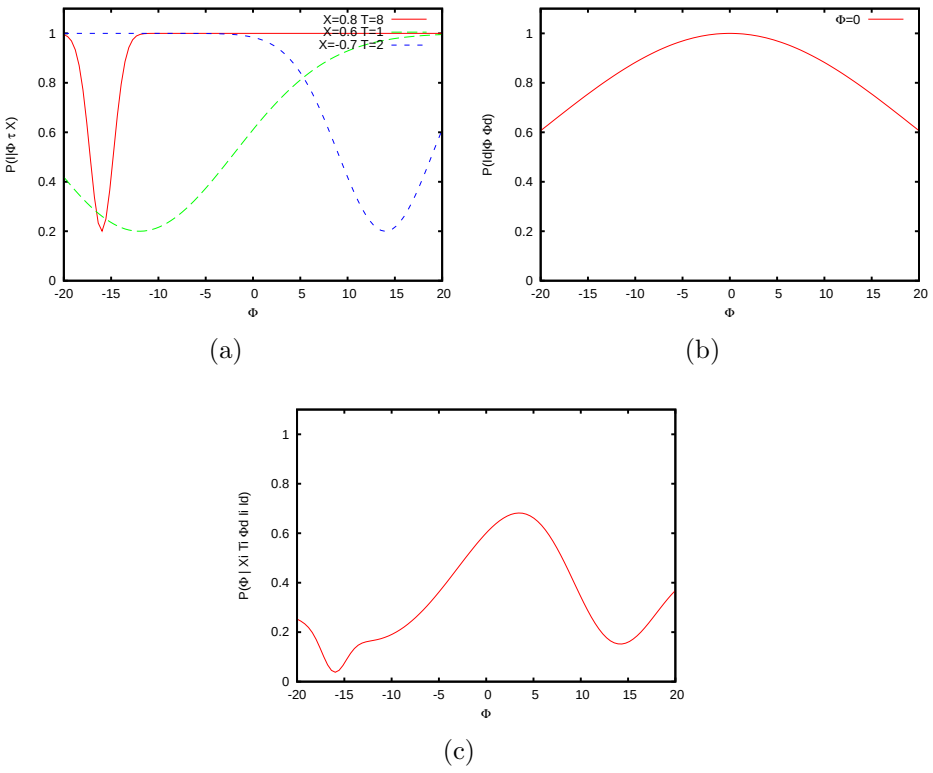


FIG. 6.18 – Exemple de distributions associées aux obstacles (a), au suivi de consigne (b) et de la fusion (c). Dans cet exemple, 3 obstacles sont détectés avec un faible TTC. Avec cette configuration, le robot va tourner à gauche (Φ positif) pour éviter les deux obstacles situés à droite mais sans braquer trop pour éviter l'obstacle de gauche.

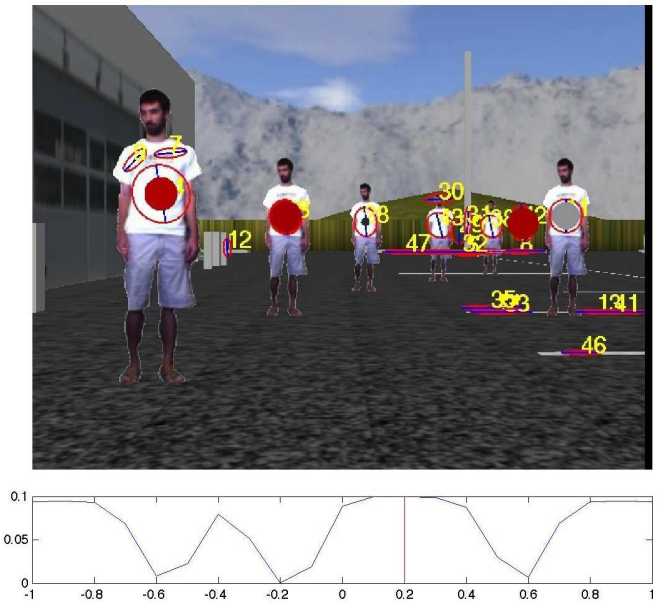


FIG. 6.19 – Exemple de distribution sur la commande de l'angle de braquage en condition réelle. Les disques rouges correspondent aux segments de crêtes suivis dont le TTC est inférieurs à 10s. La commande la plus probable se situe légèrement à droite, ce qui va faire tourner le robot de ce côté.

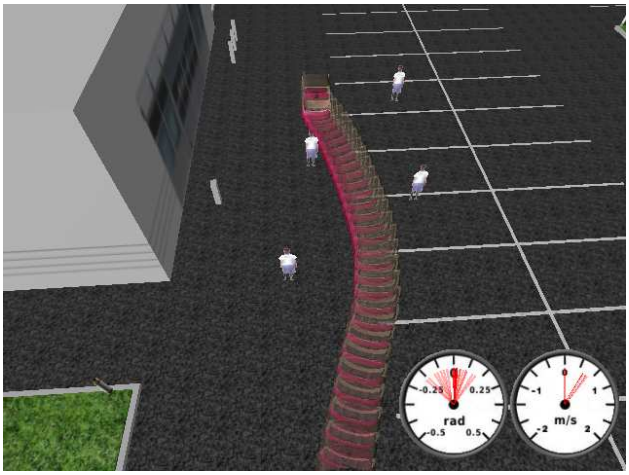


FIG. 6.20 – Exemple de navigation réactive avec évitement d'obstacles en simulation. La trainée semi-transparente représente la trajectoire du robot.

6.4 Conclusion

Dans ce chapitre consacré aux expérimentations et à l'évaluation de nos algorithmes, nous avons tout d'abord décrit la plateforme expérimentale utilisée ainsi que différentes applications développées dans le cadre de nos expérimentations, mais pouvant être utilisées plus largement dans le cadre de la robotique et du traitement d'image. Nous avons également proposé une implantation sur processeur graphique des différents algorithmes utilisés, cela permet d'accélérer grandement la quasi-totalité des tâches de traitement en utilisant pleinement l'architecture parallèle de ces processeurs.

Ensuite, nous avons évalué et validé expérimentalement les différentes étapes de détection, de suivi et d'évaluation du temps avant collision des obstacles dans le cadre de la navigation visuelle. Nous avons ainsi étudié la répétabilité du détecteur de segments de crête qui est comparable aux détecteurs de points d'intérêts classiques mais permettant de mieux décrire une scène dans un environnement urbain. L'algorithme de suivi des segments s'est montré efficace dans le cas d'objets bien contrastés et relativement peu mobiles dans l'image. Nous avons aussi constaté une relation entre la fiabilité du suivi et l'échelle des segments, les segments à grande échelle étant plus stables que dans les petites échelles.

Enfin, nous avons réalisé deux applications de navigation. La première application consiste à stopper le véhicule avant une collision avec un obstacle prédéfini et la seconde à effectuer une tâche de navigation réactive avec évitement d'obstacles. Ces deux applications ont validé notre approche fondée sur les segments de crêtes et l'estimation du temps avant collision par variation de l'échelle caractéristique pour la détection et l'évitement d'obstacles.

Chapitre 7

Conclusion

Dans ce manuscrit, nous nous sommes intéressés au problème de la détection d'obstacle dans un environnement dynamique de type urbain. Pour cela, nous avons proposé une méthode visuelle de détection et de suivi de structures particulières adaptées aux éléments présents dans ce milieu. Ces structures correspondent à des régions contrastées et allongées dans l'espace image, appelées segments de crête. Ce détecteur a été conçu en utilisant des invariants visuels fondés sur la notion d'espace d'échelle, de telle sorte que les structures soient détectées quelle que soit leur orientation et leur taille dans l'image.

Dans le but d'estimer les paramètres dynamiques des éléments détectés, nous avons mis au point un algorithme de suivi utilisant un filtre bayésien, et plus précisément un filtre à particules. Ce type de filtre a été choisi de manière à être le plus réactif possible par rapport à des variations de mouvement rapides. Dans cette étude, nous avons défini un modèle dynamique de déplacement des points ainsi qu'un modèle d'observation fondé à la fois sur la fonction de détection des segments de crêtes et sur l'apparence locale des segments. Nous avons également proposé une méthode de suivi multi cibles permettant de suivre simultanément un certain nombre d'éléments.

Enfin, pour sélectionner les éléments à risques dans l'image, nous avons proposé une nouvelle méthode d'estimation du temps avant collision utilisant exclusivement la variation des échelles caractéristiques. Cette méthode peut être appliquée en utilisant les données du tracker de cible mentionné précédemment pour estimer le temps avant collision de chacune des cibles suivies, ce qui permet d'extraire les zones à risques dans l'image.

Pour une utilisation en temps réel (à fréquence vidéo), nous avons proposé une implantation sur carte graphique de l'ensemble de nos algorithmes qui permet d'accélérer grandement les calculs.

7.1 Résultats

Pour valider notre approche, nous avons tout d'abord évalué expérimentalement les deux principales étapes du processus d'identification des obstacles qui sont la détection et le suivi des segments de crêtes. Nous avons montré que des segments de crêtes sont présents sur la quasi totalité des éléments présents dans l'environnement urbain, ce qui rend le détecteur bien adapté à cet environnement sans être spécifique à un type d'objet. De plus, nous avons prouvé que ce détecteur est invariant aux transformations affines (rotation et changement d'échelle). La méthode de suivi des cibles a été évaluée en condition réelle en utilisant une caméra embarquée sur un véhicule. Ces expérimentations nous ont permis de valider cet algorithme pour la majorité des cibles suivies mais nous avons également pu remarquer que le rapport entre l'échelle et la vitesse de la cible joue un rôle significatif sur la fiabilité du suivi (la fiabilité étant meilleure lorsque ce rapport est élevé).

Ensuite, nous avons validé la méthode d'estimation des risques par variation des échelles caractéristiques sur deux applications de navigation visuelles. Les résultats obtenus ont montré que notre méthode permet de prédire précisément les collisions et peut être utilisée pour contrôler l'arrêt ou la direction du véhicule.

Les points forts et les limites de notre méthode sont principalement liés au domaine de fonctionnement du détecteur et du suivi des segments de crête. Notre méthode donne de bons résultats lorsque les obstacles présentent une ou plusieurs formes allongées et contrastées comme c'est le cas de la pluparts des éléments présents dans l'environnement urbain (piétons, voitures, poteaux, *etc...*). En revanche, la méthode proposée est mal adaptée à un environnement dans lesquels les obstacles sont peu contrastés, comme par exemple un rocher sur un chemin de terre, ou lorsqu'ils sont très texturés. En effet dans ce cas, le suivi des structures est difficile à effectuer du fait de la multitude des crêtes pouvant être détectées et confondues.

7.2 Perspectives

Parmi les perspectives envisageables pour prolonger notre travail, nous pouvons distinguer deux catégories, la première correspondant aux améliorations et la seconde à l'utilisation possible de nos travaux.

Parmi les améliorations possibles, nous proposons les extensions suivantes :

Suppression des fausses détections : parmi les objets détectés et suivis

par notre méthode, un certain nombre d'objets correspondent à des éléments ne constituant pas des obstacles. Etant donné que notre détecteur est sensible à toutes les zones de fort contraste, on détecte par exemple les marques du sol ainsi que les ombres projetées. Une méthode de classification (par l'apparence ou le mouvement par exemple) pourrait se révéler très utile pour supprimer ces éléments.

Localisation spatiale des obstacles : bien que le TTC fournisse une information précieuse sur la dangerosité des obstacles, cette information ne suffit pas à elle toute seule pour éviter les obstacles. La position des obstacles dans l'image permet de les situer approximativement par rapport à la caméra (à droite ou à gauche) mais il serait intéressant de localiser plus précisément ces obstacles par rapport au robot.

Prise en compte de la vitesse : la vitesse des points dans l'image peut apporter de nombreuses indications sur le danger de collision complémentaire au temps avant collision. Par exemple, un objet fixe dans l'image mais qui grossit vite représente plus de danger qu'un objet qui grossit mais en se décalant dans l'image.

Ensuite, nous pouvons mettre en avant plusieurs points de notre méthode pouvant être utilisés dans d'autres domaines que la détection d'obstacle.

Utilisation des segments de crêtes : le détecteur et l'algorithme de suivi des segments de crêtes pourraient être utilisés dans d'autres types d'application telle que la reconnaissance d'objets en étant utilisés soit comme des points d'intérêts, soit comme des descripteurs de structure.

SLAM visuel : dans le domaine de la localisation et cartographie simultanées plus connu sous le nom de *SLAM* en anglais, les segments de crêtes pourraient également être utilisés sous forme d'amers visuels. Le premier intérêt de tels amers est le fait qu'ils soient localisés aux centre des objets "physiques". Ensuite, la dimensionalité de ce type d'amer permet d'utiliser plus d'information qu'avec un simple point d'intérêt. On peut ainsi utiliser notre détecteur avec des algorithmes de SLAM visuels utilisant des segments de lignes, tels que [ED06, LL07]. Enfin, de la même manière que nous estimons le temps avant collision, il est possible, connaissant la vitesse de la caméra, de déterminer la distance des amers fixes. Cela peut permettre d'initialiser facilement la position des amers, même sans avoir de mouvement de parallaxe, ce qui est impossible avec les algorithmes actuels.

Annexe A

Implémentation sur processeur graphique

A.1 Architecture des GPU

Les processeurs graphiques ont pour fonction principale d'effectuer le rendu de points et polygones. Ils reçoivent donc en entrée la géométrie des objets à afficher, les textures de ces objets ainsi que de nombreux paramètres tels que les éclairages, la position de la caméra, etc. Ensuite le processeur se charge de projeter les objets à l'écran (ou dans une image). Du traitement de la géométrie à l'affichage dans l'image finale, on peut diviser le pipeline graphique en quatre parties :

- **Le vertex pipeline :** C'est la partie qui transforme la géométrie en fonction de la position de la caméra et des transformations appliquées à chaque vertex. Cette partie est programmable, ce qui signifie qu'on peut modifier la manière dont les vertex sont transformés ou projetés dans l'image. Ces programmes sont appelés *vertex shaders*.
- **La rasterization :** C'est la partie qui transforme les primitives géométriques en un ensemble de pixels dans l'image finale. Cette partie gère aussi certaines fonctionalités telles que le *culling* – qui consiste à supprimer les faces qui ne sont pas orientées vers l'observateur – ou le *clipping* qui permet de supprimer les pixels non visibles (hors du champ de vue par exemple). Cette partie n'est pas programmable, on peut uniquement activer ou désactiver certaines fonctionalités.
- **Le pixel pipeline :** C'est la partie qui calcule la couleur de sortie de chaque pixel en fonction de la couleur des vertex, des textures, et des éclairages. Cette partie est programmable, on peut donc modifier la manière dont la couleur est calculée. Les programmes utilisés à ce

niveau sont appelés *fragment shaders*.

– **Composition :** C'est la partie qui gère la façon dont les pixels sont superposés (par exemple il est possible de tester la profondeur des pixels pour n'afficher que le plus proche pixel en cas de superposition, ou de gérer la transparence en mélangeant la couleur des pixels).

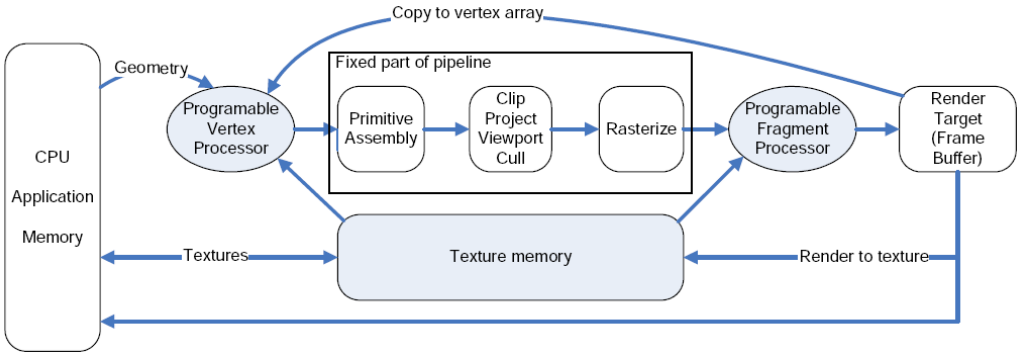


FIG. A.1 – Pipeline graphique sur les processeurs graphiques modernes.

Au niveau matériel, les processeurs graphique sont constitués de nombreuses unités de calculs de types processeurs de flux *SIMD* (Single Instruction Multiple Data), ce qui signifie que les mêmes instructions sont exécutées en parallèle sur des données multiples. En outre, des unités de textures sont partagées entre les différents processeurs de flux pour accéder aux textures avec la possibilité d'effectuer des filtrages (bilinéaire ou trilinéaire).

En pratique, cette architecture est très efficace pour appliquer les mêmes programmes sur un nombre important de données et lorsque ces données sont bien agencées en mémoire (par exemple sous forme de tableau).

A.2 Outils de développement

– **OpenGL / GLSL** : cette librairie fournie une API multi-plateforme pour la conception d'application graphique 2D et 3D. Cette librairie permet d'exploiter le processeur graphique en utilisant le pipeline fixe (rendu graphique simple), mais aussi les parties programmables en utilisant le langage de programmation GLSL (OpenGL Shading Language). En outre, une version de cette librairie (OpenGL ES) a été conçue pour les applications embarquées, ce qui peut être intéressant en robotique mobile.

- **DirectX** : cette librairie est en fait une collection de bibliothèques permettant la programmation d'application multimédia. En particulier, Direct 3D est utilisé pour le graphisme 3D et permet d'utiliser l'accélération 3D des processeurs graphiques. Cependant, cette librairie n'est utilisable que sur les plateformes Microsoft (systèmes d'exploitation Windows et Xbox), ce qui rend cette librairie peu exploitable.
- **CUDA** : cette technologie regroupe une librairie, un runtime et un driver permettant d'exploiter directement le processeur graphique sans passer par le pipeline graphique en exécutant des programmes sur un certain nombre de threads. Avec cette technologie, il est possible d'utiliser plus de fonctionalités bas niveau qu'avec les autres librairies plus haut niveau, et on s'affranchit aussi de certaines contraintes liées au pipeline graphique. Les programmes peuvent par exemple écrire à n'importe quelle adresse mémoire et peuvent utiliser explicitement la mémoire partagée des blocs processeurs. Toutefois, cette librairie est pour l'instant réservée aux architectures nVidia et est donc relativement peu portable.

Pour des raisons de portabilité, nous avons choisi d'utiliser la librairie OpenGL pour développer nos applications. Les instructions sont écrites en utilisant le langage GLSL. Les entrées et les sorties sont passées sous formes de textures.

A.3 Implémentation des algorithmes

A.3.1 Filtrage et construction de l'espace d'échelle

Comme on l'a vu en 3.2.5 la construction de l'espace d'échelle pyramidal comporte 2 étapes qui sont répétées pour chaque niveau d'échelle : filtrage et sous-échantillonage. Le filtrage gaussien s'effectue en stockant l'image dans une texture et en exécutant un programme sur chaque pixel (fragment shader). Etant donné que les noyaux sont séparables, on effectue deux passes. Dans chaque passe, le programme va exécuter la convolution du noyaux vertical ou horizontal en sommant les valeurs des pixels voisins pondérées par les coefficient du noyaux. Pour le sous-échantillonage, on fait un rendu direct de la texture en divisant la taille par deux.

Le calcul des dérivées correspond à un filtrage avec un noyau séparable, on peut donc appliquer la même méthode que pour la convolution gaussienne en modifiant le noyau et en appliquant le programme à chaque niveau de la pyramide.

Ensuite, pour accéder rapidement à n'importe quel point de l'espace d'échelle, on transforme l'ensemble des textures en une texture 3D en effectuant un simple rendu de texture pour chaque plan de la texture 3D.

A.3.2 Filtre à particules

Le filtre à particules est assez facile à paralléliser puisque les mêmes opérations sont effectuées sur un grand nombre de particules. Pour effectuer le suivi multi-cibles, il est nécessaire de maintenir à jour l'état de plusieurs filtres, chaque filtre étant représenté par un grand nombre de particules. Nous allons représenter l'état du filtre par une texture 2D où chaque ligne représente un filtre à particules et chaque pixel représente une particule. L'état d'une particule comprend 9 paramètres (position et vitesse du centre (3D+3D) + demi-segment (3D)) il faut ajouter à cela le poids de la particule, ce qui fait 10 paramètres. Etant donné qu'un texture OpenGL ne peut être composée que de 4 canaux au maximum, nous avons besoin d'au moins 3 textures, nous en utilisons 4 en réalité pour séparer le poids et l'état des particules.

Pour appliquer le modèle de transition, on associe un programme à chaque particule (ou pixel). Le programme lit l'état de la particule, applique le modèle de transition (intégration avec la vitesse + ajout de bruit) et écrit le nouvel état dans la texture (voir Figure A.2).

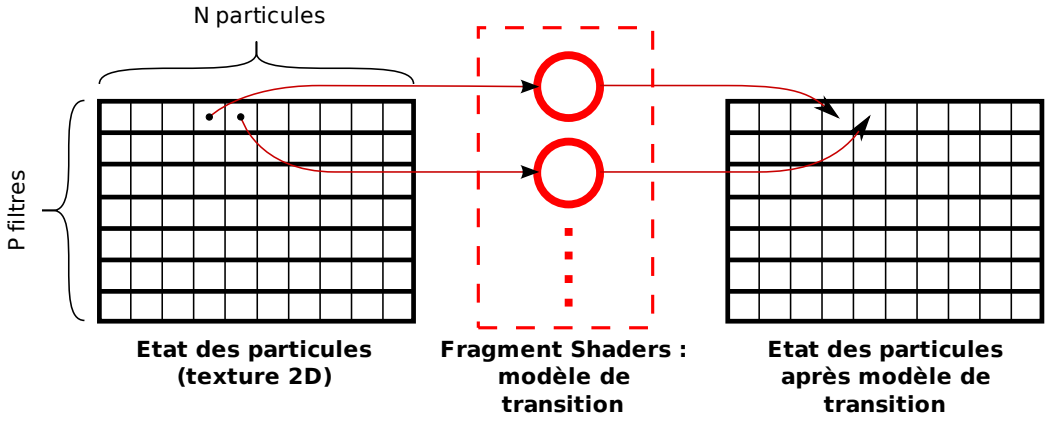


FIG. A.2 – Application du modèle de transition sur GPU. Le modèle de transition est appliqué en parallèle sur l'ensemble des particules de tous les filtres à l'aide de fragment shaders.

Pour le modèle d'observation, on associe de même un programme à chaque particule, le programme lit l'état de la particule et calcule la probabilité

d'observation à l'aide de l'espace d'échelle laplacien et des descripteurs associés à chaque cible (voir Figure A.3).

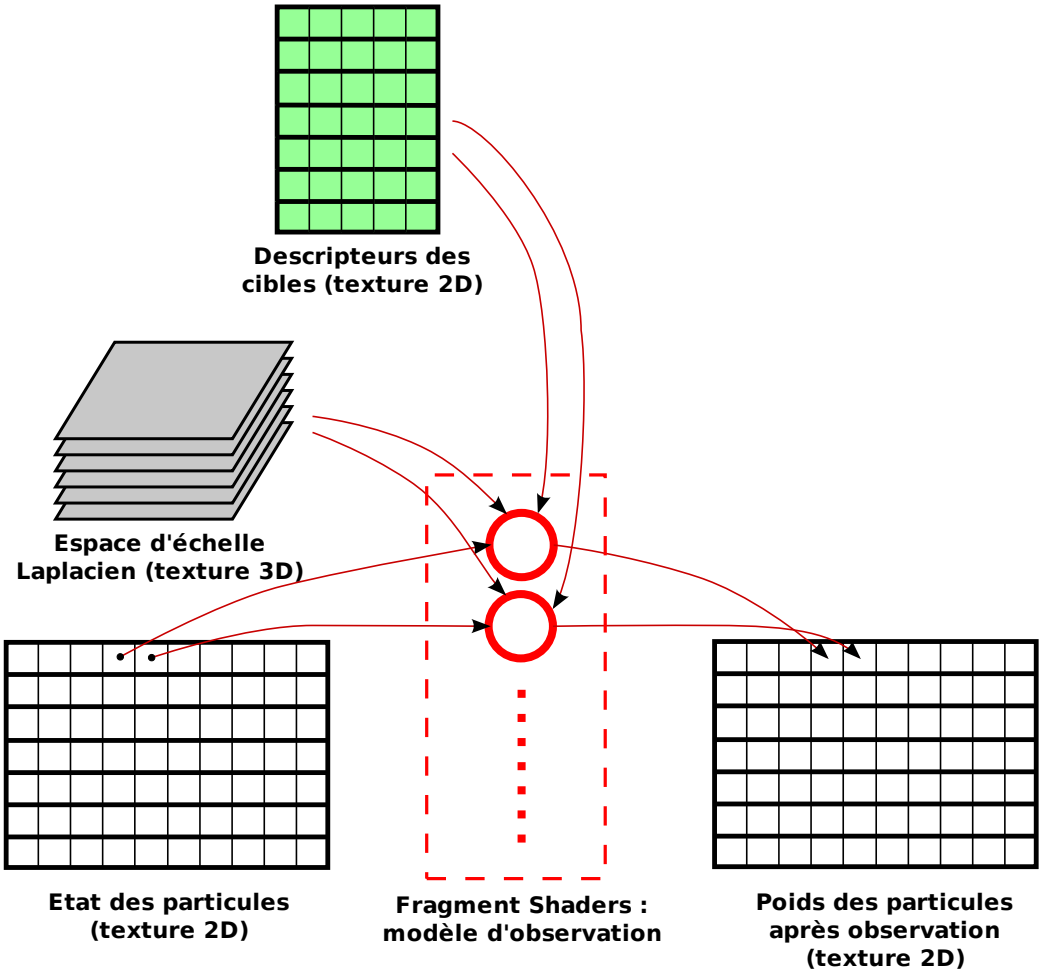


FIG. A.3 – Application du modèle d'observation sur GPU. Le modèle d'observation est appliqué en parallèle sur l'ensemble des particules de tous les filtres à l'aide de fragment shaders.

Ensuite, pour extraire l'état moyen de chaque filtre, le processus n'est pas directement parallélisable. En effet, il est nécessaire de calculer une somme sur toutes les particules du filtre. Il est possible d'utiliser un algorithme ayant un coût logarithmique, mais il faut dans ce cas effectuer plusieurs passes, et dans ce cas le gain en temps de calcul n'est pas forcément important. Nous allons alors utiliser un algorithme séquentiel pour calculer les moyennes, mais paralléliser cet algorithme sur l'ensemble des filtres à

particules. Ainsi, on lance un programme par filtre à particules, chaque programme lit l'ensemble des particules et calcule l'état moyen, ainsi que certain indicateurs statistiques tels que l'indice de dégénérescence (n_{eff}) et la probabilité maximale des particules w_{max} (voir Figure A.4).

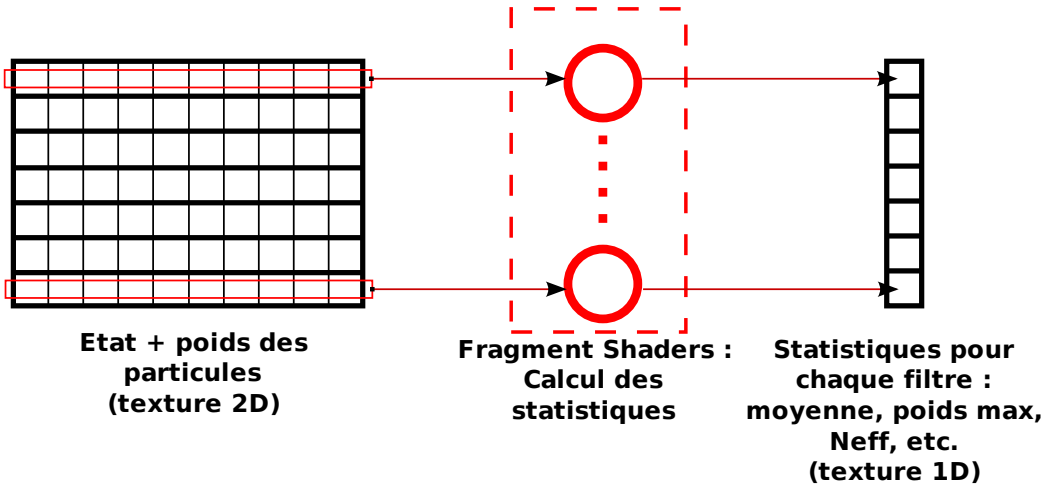


FIG. A.4 – Calcul des statistiques sur GPU. Un fragment shader par filtre est utilisé, chaque shader lit l'ensemble des particules associé au filtre pour calculer la moyenne, le poids maximum, l'indicateur de dégénérescence, etc.

Pour le resampling et la normalisation du filtre, on lance un programme par particule, chaque programme lit la variable n_{eff} du filtre correspondant à la particule, si le resampling est nécessaire, alors la particule est remplacée par une particule choisie par l'algorithme du rejet (voir Algorithme 4) et son poids est mis à 1. Sinon le poids est normalisé en fonction du poids maximal des particules w_{max} calculé précédemment (voir Figure A.5).

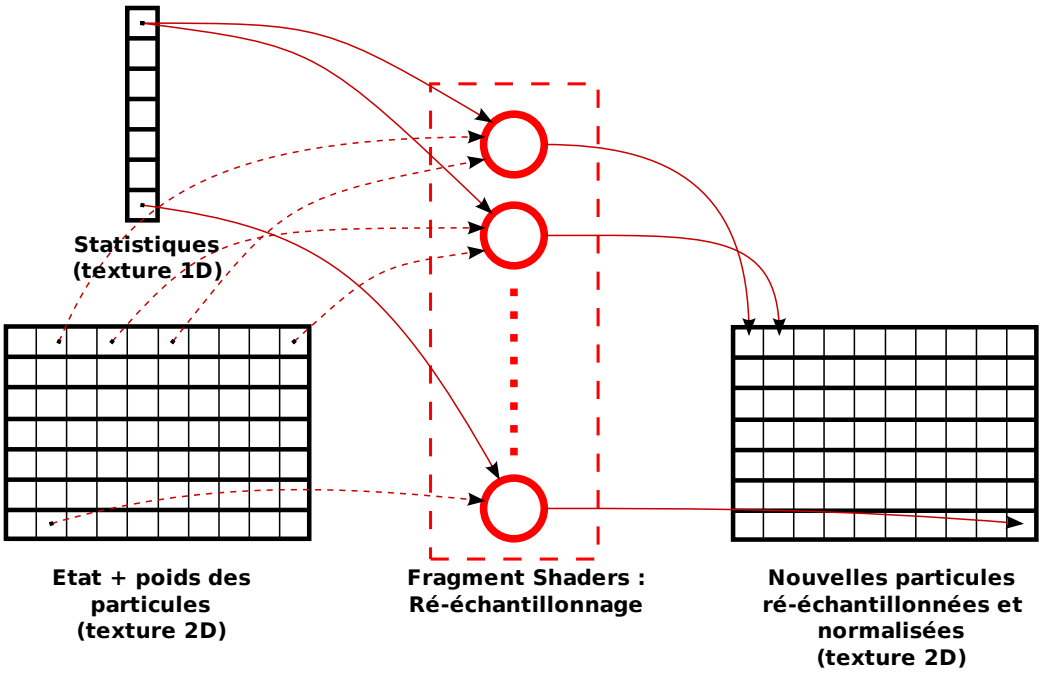


FIG. A.5 – Ré-échantillonage des particules sur GPU : un fragment shader par particule est utilisé. Chaque shader lit les données statistiques du filtre correspondant à la particule, si le ré-échantillonage est nécessaire, le programme choisit aléatoirement une nouvelle particule parmi toute les particules du filtre en utilisant la méthode du rejet. Le programme écrit ensuite l'état de cette particule avec un poids normalisé dans la texture de sortie.

Bibliographie

[AAT07] G. Alenya, M. Alberich, and C. Torras. Depth from the visual motion of a planar target induced by zooming. *IEEE International Conference on Robotics and Automation*, pages 4727–4732, April 2007.

[AMGC02] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2) :174–188, Feb 2002.

[ASB94] S. Ayer, P. Schroeter, and J. Bigün. Segmentation of moving objects by robust motion parameter estimation over multiple frames. In *European conference on Computer Vision*, pages 316–327, 1994.

[AT08] G. Alenyà and C. Torras. *Artificial Intelligence Research and Development*, chapter Monocular object pose computation with the foveal-peripheral camera of the humanoid robot Armar-III. IOS Press, 2008.

[BBC+02] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi, and M. Porta. Artificial vision in road vehicles. In *IEEE Special Issue on Technology and Tools for Visual Perception*, volume 7, pages 1258–1271, july 2002.

[BBF00] M. Bertozzi, A. Broggi, and A. Fascioli. Vision-based intelligent vehicles : state of the art and perspectives. *Robotics and Autonomous Systems*, 32 :1–16, 2000.

[BBG+03] M. Bertozzi, A. Broggi, P. Grisleri, T. Graf, and M. Meinecke. Pedestrian detection in infrared images. *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 662–667, June 2003.

[BCA04] A. Broggi, P. Cerri, and P. C. Antonellop. Multi-resolution vehicle detection using artificial vision. In *Proceedings of the*

IEEE Intelligent Vehicles Symposium, pages 310–314, Parma, Italy, June 2004.

[BFC+04] A. Broggi, A. Fascioli, M. Carletti, T. Graf, and M. Meinecke. A multi-resolution approach for infrared vision-based pedestrian detection. *Intelligent Vehicles Symposium*, pages 7–12, June 2004.

[Bra07] C. Braillon. *Détection d'obstacles par fusion de flux optique et stéréovision dans des grilles d'occupation*. PhD thesis, Institut National Polytechnique de Grenoble, 2007.

[BST75] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 1975.

[BWBD85] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Transactions on pattern analysis and machine intelligence*, 8(1) :26–33, 1985.

[BWF+05] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnorr. Variational optical flow computation in real time. *IEEE Transactions on Image Processing*, 14(5) :608–615, May 2005.

[BWS05] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck : combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3) :211–231, 2005.

[Cam94] T.A. Camus. *Real-time optical flow*. PhD thesis, Brown university, Providence, RI 02912, USA, September 1994.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6) :679–698, 1986.

[CCHH96] T. Camus, D. Coombs, M. Herman, and Tsai-Hong Hong. Real-time single-workstation obstacle avoidance using only wide-field flow divergence. *Proceedings of the 13th International Conference on Pattern Recognition*, 3 :323–330, August 1996.

[CHDLEA04] J. Collado, C. Hilario, A. De La Escalera, and J.M. Armingol. Model based vehicle detection for intelligent vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 572–577, Parma, Italy, June 2004.

[CHHN98] D. Coombs, M. Herman, Tsai-Hong Hong, and M. Nashman. Real-time obstacle avoidance using central flow divergence, and peripheral flow. *IEEE Transactions on Robotics and Automation*, 14(1) :49–59, Feb 1998.

[CLFK01] Robert Collins, Alan Lipton, Hironobu Fujiyoshi, and Takeo Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10) :1456 – 1477, October 2001.

[CR03] J.L. Crowley and O. Riff. Fast computation of scale normalised gaussian receptive fields. *Scale Space Methods in Computer Vision*, 2695 :584–598, 2003.

[Cro89] J.L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. *IEEE International Conference on Robotics and Automation*, 2 :674–680, May 1989.

[ED00] D. Elgammal, A. Harwood and L. Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision*, pages 751–767, 2000.

[ED06] E. Eade and T. Drummond. Edge landmarks in monocular slam. In *British Machine Vision Conference*, Edinburgh, UK, September 2006.

[EKB98] C. Eveland, K. Konolige, and R.C. Bolles. Background modeling for segmentation of video-rate stereo sequences. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 266–271, June 1998.

[EW00] A. Ewald and V. Willhoeft. Laser scanners for obstacle detection in automotive applications. *Intelligent Vehicles Symposium*, pages 682–687, 2000.

[FWSB95] J. Ferryman, A. Worrall, G. Sullivan, and K. Backer. A generic deformable model for vehicle recognition. In *BMVC*, volume 1, pages 127–136, Birmingham, United Kingdom, September 1995.

[Gav00] D. M. Gavrila. Pedestrian detection from a moving vehicle. In *Lecture Notes In Computer Science, Proceedings of the 6th European Conference on Computer Vision*, pages 37–49, 2000.

[HDS04] C. Hoffmann, T. Dang, and C. Stiller. Vehicle detection fusing 2d visual features. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 280–285, Parma, Italy, June 2004.

[HS81] B.K.P. Horn and B.G. Shunk. Determining optical flow. In *Artificial Intelligence*, volume 17, pages 185–203, 1981.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. pages 189–192, Manchester, 1988.

[HZ00] R. Hartley and A. Zisserman. *Multiple View Geometry In Computer Vision*. Cambridge, 2000.

[IB98] M. Isard and A. Blake. Condensation—conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1) :5–28, August 1998.

[JSS02] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. *Motion and Video Computing*, pages 22–27, December 2002.

[JU97] S.J. Julier and J.K. Uhlmann. New extension of the kalman filter to nonlinear systems. volume 3068, pages 182–193. SPIE, 1997.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 1960.

[KK03] Q. Ke and T. Kanade. Transforming camera geometry to a virtual downward-looking camera : robust ego-motion estimation and ground-layer detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1 :390–397, June 2003.

[Koe84] Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5) :363–370, 1984.

[KPBM03] C. Koike, C. Pradalier, P. Bessiere, and E. Mazer. Proscriptive bayesian programming application for collision avoidance. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1 :394–399, Oct. 2003.

[Kru99] K. Kruger. Robust real-time ground plane motion compensation from a moving vehicle. *Machine Vision and Applications*, 11(4) :203–212, 1999.

[LAT02] R. Labayrade, D. Aubert, and J.-P. Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. *Intelligent Vehicle Symposium*, 2 :646–651, June 2002.

[Lee76] David N. Lee. A theory of visual control of braking based on information about time-to-collision. *Perception*, 1976.

[Lee81] David N. Lee. Plummeting gannets : a paradigm of ecological optics. *Nature*, 293, September 1981.

[Lin90] Tony Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3) :234–254, 1990.

[Lin93] Tony Lindeberg. Discrete derivative approximations with scale-space properties : A basis for low-level feature extraction. *Journal of Mathematical Imaging and Vision*, 3(4) :349–376, November 1993.

[Lin94] Tony Lindeberg. *Scale-space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[Lin98] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2) :79–116, 1998.

[LK81] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. DARPA IU Workshop*, pages 121–130, 1981.

[LL07] T. Lemaire and S. Lacroix. Monocular-vision based slam using line segments. *IEEE International Conference on Robotics and Automation*, pages 2791–2796, April 2007.

[Low99] D. G. Lowe. Object recognition from local scale-invariant feature. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

[Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, volume 60, pages 91–110, 2004.

[LYR92] David N. Lee, David S. Young, and Dennis Rewt. How do somersaulters land on their feet ? *Journal of Experimental Psychology : Human Perception and Performance*, 18(4) :1195–1202, November 1992.

[MCD00] F. Marmoiton, F. Collange, and J.P. Dérutin. Location and relative speed estimation of vehicles by monocular vision. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 227–232, 2000.

[MP04] A. Mittal and N. Paragios. Motion-based background subtraction using adaptive kernel density estimation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2 :302–309, June 2004.

[MS01] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. *ICCV*, 01 :525, 2001.

[NBCL06] A. Nègre, C. Braillon, J. L. Crowley, and C. Laugier. Real-time Time-To-Collision from variation of Intrinsic Scale. In *Proc. of the Int. Symp. on Experimental Robotics*, Rio de Janeiro, Brazil, July 2006.

[NCL08] A. Nègre, J. L. Crowley, and C. Laugier. Scale invariant segment detection and tracking. In *Proc. of the Int. Symp. on Experimental Robotics*, Athens, Greece, July 2008.

[ND02] H. Nanda and L. Davis. Probabilistic template based pedestrian detection in infrared videos. In *Intelligent Vehicle Symposium*, volume 1, pages 15–20, june 2002.

[PP99] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. *International Conference on Image Processing*, 4 :35–39, 1999.

[RB78] D. Regan and K.I. Beverly. Looming detectors in the human visual pathways. *Vision Research*, 18 :415–421, 1978.

[SCG62] William Schiff, James A. Caviness, and James J. Gibson. Persistent fear responses in rhesus monkeys to the optical stimulus of "looming". *Science*, 136(3520) :982–983, 1962.

[SF98] H. Sun and B.J. Frost. Computation of different optical variables of looming objects in pigeon nucleus rotundus neurons. *Nature Neuroscience*, 1(4) :296–303, 1998.

[SFN+04] M.A. Sotelo, D. Fernandez, J.E. Naranjo, C. Gonzalez, R. Garcia, T. de Pedro, and J. Reviejo. Vision-based adaptive cruise control for intelligent road vehicles. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1 :64–69, Sept.-2 Oct. 2004.

[SMB00] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2) :151–172, June 2000.

[SMS03] G.P. Stein, O. Mano, and A. Shashua. Vision-based acc with a single camera : bounds on range and range rate accuracy. *Intelligent Vehicles Symposium*, pages 120–125, June 2003.

[SO06] A. Seki and M. Okutomi. Robust obstacle detection in general road environment based on road extraction and pose estimation. *Intelligent Vehicles Symposium*, pages 437–444, June 2006.

[TL04] T.T.H. Tran and A. Lux. A method for ridge extraction. *Asian Conference on Computer Vision*, 2004.

[Tor95] P. Torr. *Motion Segmentation and Outlier Detection*. PhD thesis, Department of Engineering Science, University of Oxford, 1995.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1 :511–518, 2001.

[VJS05] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2) :153–161, July 2005.

[VTV98] L. J. V. Vliet, Young I. T., and P. W. Verbeek. Recursive gaussian derivative filters. *ICPR*, 1 :509, 1998.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[Wit83] A. P. Witkin. Scale-space filtering. pages 1019–1022, 1983.

[XLF05] F. Xu, X. Liu, and K. Fujimura. Pedestrian detection and tracking with night vision. *IEEE Transactions on Intelligent Transportation Systems*, 6(1) :63–71, March 2005.

[ZT00] L. Zhao and C.E. Thorpe. Stereo- and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3) :148–154, Sep 2000.