REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE
Département d'automatique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

مخبر التحكم في العمليات الصناعية
Laboratoire de Commande des Processus

MEMOIRE

De Magister Génie Electrique

Option :Robotique et Productique

Présenté par :

AIB Abdelghani

Ingénieur d'Etat en Automatique de ENP- Alger

Thème

Etude et conception sur FPGA d'algorithmes de vision pour robot mobile

Soutenu publiquement devant le jury composé de :

Président:	CHEKIREB Hachemi	Professeur, ENP
Directeur :	LARBES Cherif	Professeur, ENP
Examinateurs :	AIT CHEIKH Mohamed Salah	Maitre de Conférences A, ENP
	ILLOUL Rachid	Maitre de Conférences A, ENP
Invité :	TADJINE Mohamed	Professeur, ENP

Année universitaire 2011/2012

Ecole Nationale Polytechnique, 10, AV. Hassen Badi, El-Harrach, Algérie

Remerciements

Quelques lignes ne pourront jamais exprimer la reconnaissance que nous éprouvons envers tous ceux qui, de pris ou de loin, ont contribué, par leurs conseils, leurs encouragements ou leurs amitiés à l'aboutissement de ce travail.

Mes vifs remerciements accompagnés de toute ma gratitude vont tout d'abord à mon promoteur M. C. LARBES Pour m'avoir proposé ce sujet, pour les conseils qu'il n'a cessé de me prodiguer et surtout pour la confiance qu'elle m'a accordé pour la réalisation de ce projet.

Ma reconnaissance va à tous mes enseignants à département de l'automatique et à l'ENP, en général.

Dédicaces

Arrivé à ce stade, n'est que le fruit du milieu familial qui m'est propice, le fruit de l'équilibre et du sacrifice de mes Parents, Ces êtres chers que Dieu les protège.

Qu'il m'est agréable en ce moment de partager ce bonheur avec eux.

Je dédie ce modeste travail à mes chers Parents qui ont été de tout temps, les plus proches , qui n'ont jamais ménagé leurs efforts, leurs encouragements et leur soutien avec abnégation et patience

Sans oublier mes Grands Parents.

A ma femme et ma fille maram.

A mon frère et mes sœurs.

A mes Oncles, Tantes et cousins et alliés de la famille.

En ce moment je ne peux oublier.

A l'ensemble des amis que j'ai connu pendant mes études et à ceux qui ont prodigué leurs vifs conseils, encouragements et témoigné de leur amitié.

ملخص:

يندرج العمل المعروض في هذه الأطروحة في إطار الرؤية في الروبوتيك المتحركة، و يتمثل في الإخضاع البصري للروبوت المتحرك من أجل اكتشاف الحواجز باستعمال كاميرا ستيريوسكوبية. لتجسيد هذا العمل، قمنا بتثبيت خوارزمية خاصة بالرؤية من أجل روبوت متحرك على بطاقة الكترونية، بطاقة قابلة للبرمجة في حالتنا.

مفاتيح :الروبوتيك المتحركة، الرؤية الستيريوسكوبية ، الإخضاع البصري، اكتشاف الحواجز ،البطاقات القابلة للبرمجة.

Résumé :

Le travail présenté dans ce mémoire s'inscrit dans le cadre de la vision en robotiquemobile et consiste en l'asservissement visuel d'un robot mobile pour la détection d'obstacles avec une caméra stéréoscopique. Pour envisager une telle application, nous somme implémentés un algorithme de vision pour un robot mobile sur un support électronique, un circuit FPGA dans notre cas.

Mots Clés : Robotique mobile, vision stéréoscopique, asservissement visuel, détection d'obstacles, FPGA.

ABSTRACT:

The work presented in this concerns vision in mobile robotics. It consists of the visual servoing of a mobile robot for the obstacles detection with a stereoscopic camera. To consider such an application, we implement a vision algorithm for mobile robot with electronic support, the FPGA circuit of this situation.

Keywords:Mobile robotic, stereoscopic vision, visual servoing, obstacle detection, FPGA.

SOMMAIRE

CHAPITRE 4 Asservissement visuel des robots mobiles

La liste des figures

La liste des tableaux

Introduction générale

Introduction générale

La vision stéréoscopique humaine permet à l'homme de se déplacer et d'appréhender son environnement : détecter un objet, reconnaître un visage dans une scène et ressentir les émotions des gens, perceptibles à travers leurs expressions, sont quelques exemples de fonctions dévolues à la vision et quotidiennement exécutées de manière implicite. Bien que celles-ci soient réalisées avec un minimum d'effort, elles traduisent lorsqu'on tente de les analyser l'existence d'un ensemble de processus extrêmement complexes.

Dans les trois dernières décennies, les chercheurs ont essayé de doter de capacité de vision les systèmes automatisés tels les robots afin de les rendre plus intelligents et d'élargir ainsi l'éventail de leurs capacités. Cependant beaucoup de chemin reste à faire : la façon dont le cerveau humain traite l'information visuelle est très mal connue et les tâches de traitement d'image couramment exécutées par les systèmes automatisés sont très lourdes en temps de calcul. En développant les systèmes de vision, les concepteurs sont essentiellement confrontés à deux possibilités : implémenter les algorithmes de vision sous forme logicielle et les exécuter sur des processeurs standards (microprocesseur, DSP), ou les implémenter dans des architectures matérielles spécialement conçues pour l'application(ASICS). Bien que les architectures matérielles spécifiques puissent offrir des vitesses de traitement beaucoup plus élevées et puissent exploiter beaucoup plus facilement le parallélisme que l'on rencontre fréquemment dans les algorithmes de traitement d'images, celles-ci sont restées longtemps ignorées car les ressources nécessaires ne sont pas toujours disponibles à un prix raisonnable et le délai de conception est relativement long. Ces différents aspects ont contribué au choix fréquent de la première option, à savoir le développement d'algorithmes tournant le plus rapidement possible sur des processeurs standards.

Depuis quelques années, une troisième solution pour les concepteurs de systèmes de vision a vu le jour du fait de la croissance très rapide des circuits logiques programmables (FPGA) en termes de vitesse, de ressources disponibles et de performances des outils de développement. Les circuits logiques programmables ont également été utilisés avec succès dans des applications autres que celles liées au traitement d'images, telle que le filtrage numérique multivoies, la FFT, la formation de voies dans les antennes acoustiques large bandes. Ces circuits permettent aux développeurs de configurer économiquement et

rapidement leur structure interne en fonction des caractéristiques de l'algorithme de traitement choisi car ils éliminent les phases les plus lourdes et les plus coûteuses que l'on rencontre lors de la conception de circuits intégrés spécifiques tels les ASIC, à savoir leur fabrication. Ils permettent également une réduction significative des temps de mise au point puisqu'il est très facile et très rapide pour le concepteur de modifier le design, le recompiler et reprogrammer le composant.

Le but des travaux présentés dans ce mémoire est d'implémenté un algorithme de vision pour un robot mobile sur un support électronique, un circuit FPGA dans notre cas, et le deuxième objectif est d'évaluer les performances des circuits FPGA pour les applications de robotique et de vision temps réel basées sur le principe V-Disparité pour la détection des obstacles.

Ce mémoire est structuré comme suit :

Chapitre 1 : Il présente un état de l'art sur la vision stéréoscopique ainsi que l'approche de V-disparité pour la détection d'obstacle.

Chapitre 2 : Il aborde la transformée de Hough, son principe et implémentation.

Chapitre 3 : Il présente un état de l'art sur l'asservissement visuel des robots mobiles.

Chapitre 4 : Il présente les outils de développement d'un projet sur La kit RC203E tel que la carte FPGA Virtex-II, l'environnement de développement DK et le langage Handel-C.

Chapitre 5 : Ce chapitre est dédie à l'implémentation hardware de l'approche de V-disparité pour la détection d'obstacles sur un circuit FPGA.

Chapitre 6 : Dans ce dernier chapitre, on présent les résultats de simulation de comportement d'un robot mobile en présence d'un bloc de vision, la validation de l'approche de V-disparité sur un outil de développement software ainsi que les résultats de développement de l'approche de V-disparité sur la carte RC203E.

Enfin nous terminons par une conclusion générale.

La détection d'obstacles par stéréovision

1. La vision artificielle

1.1. Préambule

La vision artificielle est la science qui développe les bases algorithmiques et théoriques par lesquelles l'information utile relative à l'environnement peut être automatiquement extraite et analysée à partir d'une image, d'un ensemble d'images ou d'une séquence d'images.

Une telle information peut référer à la reconnaissance d'un objet générique, à la description tridimensionnelle d'un objet inconnu, à la position et l'orientation d'un objet observé, ou à la mesure de toute propriété spatiale d'un objet telle que la distance entre deux de ces points distincts ou le diamètre d'une section circulaire.

L'acquisition d'images est réalisée par l'intermédiaire d'une caméra vidéo dont le signal est numérisé. Une caméra vidéo comprend un système optique composé de lentilles qui forme une projection bidimensionnelle d'une scène de l'environnement observée sur un plan image photosensible (ou rétine). Chaque caméra délivre, à cadence vidéo, une copie de l'image optique reçue par sa rétine électronique.

La rétine d'une caméra est divisée, en abscisse et ordonnée, en un certain nombre de points appelés pixels. Au cours du traitement de l'image, l'usage de l'informatique implique cette division en un nombre défini mais programmable de pixels.

1.2. L'image

Une image est une représentation spatiale d'un objet, d'une scène bidimensionnelle ou tridimensionnelle, ou d'une image elle-même. Elle peut être réelle ou virtuelle telle qu'en optique expérimentale. En vision, le terme image réfère usuellement à une image enregistrée, telle que l'image vidéo, l'image digitale ou la photographie.

Une image optique monochrome est abstraitement assimilable à une fonction continue **I** de deux variables réelles u et v définie sur une région rectangulaire bornée d'un plan. La valeur de la fonction image I au point de coordonnées (u, v) du plan est dénotée I (u, v). Pour les capteurs optiques ou photographiques, la valeur I (u, v) est typiquement proportionnelle à l'énergie lumineuse radiante reçue dans la gamme de fréquences électromagnétiques à laquelle le capteur d'image est sensible, au voisinage du point de coordonnées (u, v). Dans ce cas, l'image est appelée image d'intensités ou de luminances **[NET89].**

La radiance dénote la quantité de lumière émise par une source en terme de puissance par unité de surface comprise dans un angle solide unité. La fonction d'intensités de l'image I s'apparente à la radiance de la scène et dépend de la quantité de lumière incidente, de la fraction de la lumière incidente réfléchie ainsi que de la géométrie de la réflexion de la lumière, c'est-à-dire des directions d'illumination et d'observation. Ainsi, différents points d'objets de l'environnement situés devant le système optique d'acquisition d'images présentent différentes valeurs d'intensités sur l'image, dépendant de la radiance incidente ainsi que des manières dont ils sont illuminés, dont ils réfléchissent la lumière, dont la lumière réfléchie est collectée par le système optique et enfin dont le capteur d'images répond à l'excitation lumineuse **[HAR93].**

Une image digitale ou numérique est représentée par une matrice de valeurs numériques correspondant à des valeurs d'intensités quantifiées discrètes. Dans ce cas, I forme une matrice et I (u, v) représente la valeur d'intensité à la position de l'image matérialisée par la ligne u et la colonne v de la matrice. L'unité de l'image digitale ou numérique est le pixel qui associe une position (u, v) dans le plan image et une valeur d'intensité correspondante I(u, v) **[HAR92].**

Dans le contexte des images monochromes, noir et blanc, qui nous intéresse plus particulièrement, les valeurs d'intensités associées aux pixels sont appelées niveaux de gris. Le niveau de gris d'un pixel représente une mesure de l'accumulation de la lumière collectée dans la région du plan occupée par ce pixel. Dans le contexte des images trichromes couleur rouge, vert et bleue, la représentation fonctionnelle de l'image fait intervenir 3 fonctions d'intensités IR (u, v), IV (u,v) et IB(u,v) relatives aux trois composantes primaires rouge, verte et bleue constitutives de la couleur **[MAR87].**

1.3. Les capteurs d'images

1.3.1. Principe

Une image lumineuse est la représentation d'un objet, d'une scène ou d'une image elle-même obtenue par des procédés optiques. L'image est caractérisée par la distribution spatiale de ses éclairements et c'est précisément à cette distribution qu'est liée l'information spécifique qu'elle porte.

Les capteurs d'images délivrent des signaux électriques dans lesquels se trouve transposée cette information recherchée dans l'image lumineuse originelle **[ASC91].** Les signaux électriques sont ultérieurement susceptibles de traitements électroniques, qui permettent en particulier la transmission et la reconstitution de l'image sur un écran cathodique mais également leur numérisation qui conduit à une représentation digitale des images.

Un capteur d'images est formé d'une association ordonnée d'un grand nombre de capteurs optiques sur l'ensemble desquels se trouve projetée l'image. Un capteur optique délivre un signal électrique proportionnel à l'éclairement moyen de sa surface photosensible ou photosite. En conséquence, un capteur d'images fournit un ensemble de signaux représentatifs des éclairements aux divers points de l'image. Il y a ainsi échantillonnage spatial de l'image : à chaque photosite associé à un capteur optique élémentaire correspond une fraction élémentaire de l'image dite pixel ("picture element").

Les signaux propres à chaque capteur optique doivent être collectés les uns après les autres et dans un ordre parfaitement déterminé de façon que soit connue leur position d'origine dans l'image afin que sa reconstitution soit possible. L'ordre dans lequel s'effectue cette collecte des signaux définit le mode d'acquisition de l'image ; de façon quasi-générale, l'image est balayée ligne par ligne et de haut en bas.

De par la conception du capteur d'images, l'effet photoélectrique produit en chaque photosite une accumulation de charges proportionnelle à son éclairement et à la durée constante qui sépare les collectes successives de l'information portée par le photosite. L'ensemble des charges portées par les divers photosites forme une image électrostatique qui est la transposition électrique de l'image optique. Le dispositif de lecture de l'état électrostatique de chaque photosite est complété par un circuit de conversion de charge en un courant et donc une tension proportionnelle. La succession ordonnée de ces tensions locales constitue un signal de tension analogique qui forme le signal d'image ou signal vidéo. Numérisé, ce signal conduit à la représentation informatique discrète de l'image.

1.3.2. Types de capteurs

Un capteur d'images, quel qu'il soit, comprend une surface continue ou discontinue, dont le matériau est siège d'un effet photoélectrique et sur laquelle la projection optique de l'image établit une répartition de charges qui est la transposition électrostatique de la répartition des éclairements.

C'est par la méthode et la technologie utilisées pour l'analyse de cette répartition de charges que se distinguent les principaux types de capteurs d'images : les capteurs à tube et les capteurs intégrés.

Les capteurs d'images à tube mettent à profit les techniques utilisées dans la réalisation des tubes amplificateurs à vide. L'analyse de la surface photoélectrique s'y effectue par le balayage d'un faisceau d'électrons issu d'une cathode, accéléré et concentré au moyen d'électrodes portées à des potentiels élevés, et dévié sous l'action de champs magnétiques créés par des bobines parcourues par des courants en dent de scie. De leur constitution même résultent les principaux inconvénients de ce type de capteurs : encombrement, fragilité, durée de vie limitée, tension d'alimentation élevée, puissance consommée importante.

Au sein de ces capteurs de technologie ancienne se distinguent deux classes de dispositifs que constituent les tubes à photo-émission (iconoscope, image orthicon) et les tubes à photoconduction (vidicon, plumbicon,…) **[KUN93].**

Les progrès récents de la micro-électronique ont permis la réalisation de capteurs d'images intégrés. Sur une même puce de silicium se trouvent regroupés l'ensemble des circuits nécessaires à la conversion photoélectrique et à l'analyse de l'image, à savoir :

- les photoéléments ou capteurs optiques élémentaires organisés en ligne ou en matrice, délivrant chacun une charge électrique proportionnelle à l'éclairement de son photosite et à la durée de son exposition. Parmi ces photoéléments figurent les photodiodes et les photocapacités dérivées de la technologie MOS **[KUN93].**
- des registres analogiques permettant le stockage individuel de la charge fournie par chaque photoélément ainsi que son transfert et sa distribution ordonnée nécessaire à l'analyse de l'image. Deux systèmes de transfert de charges prédominent : le registre à décalage numérique et le registre à décalage analogique **[KUN93].**
- un convertisseur charge/tension fournissant le signal d'image analogique à la chaîne de traitement en aval. Divers capteurs intégrés sont recensés, ils diffèrent par la nature des photoéléments et du système de transfert de charges adoptés : la barette de photodiodes SSPD ("Self-Scanned Photodiode Device"), les capteurs à transfert de charges CCD ("Charge Coupled Device") ou à injection de charges CID ("Charge Injection Device") dérivés tous deux de la technologie MOS **[KUN93].**

Ces types de capteurs présentent les avantages résultant de leur structure intégrée : miniaturisation, robustesse, fiabilité, tension d'alimentation faible, puissance consommée réduite.

2. Vision stéréoscopique

La vision stéréoscopique est un outil privilégié pour calculer la géométrie tridimensionnelle d'une scène observée par une ou deux caméras. La première étape du calcul est la mise en correspondance des couples d'entités (primitives) liées, extraits des deux images d'une paire stéréoscopique. L'appariement revient à établir une relation biunivoque entre primitive d'une image et leurs homologues dans l'autre image. Cette tâche aisée pour l'homme représente l'un des problèmes les plus importants et les plus difficiles de la vision robotique du fait qu'une primitive donnée dans une première image peut être associée à plusieurs primitives dans la seconde image ayant les mêmes propriétés. Se pose alors le problème du choix du bon candidat. Il est aussi possible que quelques primitives visibles dans une image soient absentes dans l'autre image à cause de la position de la caméra par rapport à la scène. Afin de lever les ambiguïtés et de permettre de dégager un ensemble d'appariements corrects, un certain nombre de contraintes ont été proposées **[BOU06]** :

- Les contraintes globales, pour résoudre le problème de l'appariement de primitives appelées loi d'unicité et loi de continuité. La première se traduit par le fait qu'une primitive dans une image a au plus un homologue dans l'autre image. La seconde s'appuie sur le fait que le monde physique est constitué de surfaces continues, elle permet à partir d'un appariement initial de prédire d'autres appariements qui viennent confirmer le premier.
- Les contraintes locales qui concernent l'orientation, la longueur de la primitive, son intensité, etc.
- La contrainte épipolaire, qui permet une fois qu'une primitive d'une image est sélectionnée, de réduire l'espace de recherche de son homologue dans l'autre image. La recherche ne se fait pas dans toute l'image, mais sur la droite épipolaire.

Deux grandes méthodes de mise en correspondance stéréoscopique. La première consiste à extraire des images gauche et droite des primitives pertinentes (segment, point d'intérêt, région…) et tenter de les apparier. L'inconvénient de ces méthodes est que la reconstruction 3D risque d'être partielle.

Dans la deuxième classe, les techniques utilisées tentent de mettre en correspondance tous les pixels des images. Autrement dit, le problème est de trouver des appariements pixel à pixel le plus dense possible. Cette dernière catégorie de méthodes semble la plus adoptée par les chercheurs du fait qu'elle permet d'obtenir un appariement dense alors que la première permet de ne mettre en relation qu'un nombre restreint d'entités dans les images.

2.1 Généralités sur la mise en correspondance

2.1.1 Les difficultés de la mise en correspondance

La mise en correspondance est indispensable à la reconstruction tridimensionnelle par triangulation. Mais c'est aussi l'une des étapes les plus difficiles du fait de son aspect intrinsèquement combinatoire. En effet, sans autre information, pour trouver la correspondante d'une primitive, il faut la comparer à toutes les primitives de l'autre image. Cette comparaison est elle-même délicate du fait des pertes d'informations occasionnées par la projection du monde 3D sur l'image. D'une part, l'information contenue dans les images est dégradée par les distorsions optiques dues au capteur que le calibrage doit modéliser et corriger. D'autre part, la seule information disponible dans les images l'information photométrique, est sensible aux changements de luminosité qui peuvent se produire d'une vue à l'autre. Ainsi la seule comparaison entre les points des deux images s'avère-t-elle peu faible. Cette variation de luminosité a aussi une influence sur l'extraction des caractéristiques d'image de plus haut niveau, donc sur leur appariement **[SOM97].**

Une autre difficulté provient des différences d'occlusion dans les deux vues. Il s'agit de portions de la scène visible depuis l'un des points de vue et invisible depuis l'autre. Ces zones d'images ne peuvent donc avoir de correspondant, malgré, parfois, leurs grandes ressemblances avec des zones candidates à l'appariement.

A ces difficultés liées au capteur stéréoscopique d'images, s'ajoutent celles provenant de la scène elle-même : un miroir ou une autre structure périodique sont des configurations qui provoquent des erreurs d'appariement ou des ambigüités qu'il est difficile de détecter ou de lever.

Ces difficultés bien connues de la mise en correspondance stéréoscopique montrent à quel point le système visuel humain est performant. En effet, la mise en correspondance de deux images par ce système ne pose aucune difficulté, qu'elles contiennent ou non des occlusions, des ombres, etc. C'est peut être ce paradoxe, entre la simplicité d'une mise en correspondance par l'humain et la complexité d'une automatisation de cette tâche, qui a suscité l'intérêt des chercheurs.

2.1.2 La disparité

Une manière de représenter le résultat d'une mise en correspondance consiste à associer à chaque pixel $P_g^{i,j}$ de l'image gauche un vecteur appelé disparité, dont ses composantes sont les coordonnées dans l'image droite du point correspondant au ce pixel.

la mise en correspondance peut donc être assimilée à la recherche d'une fonction de disparité d qui attribut une disparité à chaque pixel $P_g^{i,j}$. Dans le cas général :

$$d(P_g^{i,j}) = (u - i, v - j)^T$$

Mais comme une rectification faisant aligner les deux images est toujours faite avant la mise en correspondance, la recherche du correspondant se fera sur la même ligne et par conséquent la disparité sera réduite à $d(P_g^{i,j}) = (v - j)$ **[NAS09].**

2.2. Méthodes de la mise en correspondance

Il n'existe pas une méthode de mise en correspondance suffisamment générale pour pouvoir être appliquée à une paire d'image stéréoscopique indépendamment du contenu et du type de ces images. Il existe des méthodes principales dont voici les principales caractéristiques **[RAB00] :**

2.2.1. Mise en correspondance hiérarchique

Cette méthode a été développée pour la reconstruction de terrain à partir de deux vues aériennes. A partir d'une paire des images, on produit une hiérarchie d'images. Cette hiérarchie consiste en deux pyramides d'images. Chaque pyramide a n niveaux de précision du plus fin (ou haute résolution) au plus grossier (ou basse résolution). Une sélection de caractéristiques est réalisée dans l'image gauche à partir de l'image de résolution inférieure ce qui permet d'initialiser une zone de recherche pour trouver le correspondant du point recherché à cette résolution dans l'image droite. On recherche un correspondant à ce point dans l'image de droite au niveau de plus basse résolution et cette stratégie est poursuivie jusqu'à atteindre l'image de plus haute résolution.

2.2.2. Mise en correspondance par programmation dynamique

La programmation dynamique est une technique utile pour appareiller deux séquences tout en respectant l'ordre des éléments à l'intérieur de chaque séquence. En effet, si on suppose que la scène observée contient peu de surfaces transparentes, l'ordre des projections est le même dans les deux images. L'algorithme cherche alors le meilleur appariement entre les deux séquences parmi tous les appariements possibles. Pour trouver ce meilleur appariement, on ajoute une autre contrainte qui est la contrainte d'unicité.

2.2.3. Mise en correspondance par relaxation

Le problème de mise en correspondance peut être vu comme un problème d'étiquetage : aux points caractéristiques d'une image, on associe les points caractéristiques de l'autre image. On est alors amené à faire coopérer les points d'une image entre eux en mettant en œuvre des relations découlant des contraintes géométriques/physiques. La mise en correspondance de ces points dépend du point considéré ainsi que de son voisinage. Le terme de relaxation vient du fait que, pour chaque itération, on cherche une combinaison entre les étiquettes qui minimise un critère calculé sur l'ensemble des étiquettes.

2.2.4. Mise en correspondance par invariants

Le principe de cette méthode repose sur le fait que chaque point réel s'étant projeté dans les plans images possède des caractéristiques invariantes (par exemple l'intensité lumineuse). On caractérise ainsi chaque point de chaque image avec une liste d'invariants et à l'aide de la contrainte épipolaire, on cherche à mettre en correspondance les points qui offrent une liste d'invariants similaire.

2.2.5. Mise en correspondance par corrélation

Avec cette méthode, pour garantir de bonnes performances, la rectification et la correction des distorsions sont indispensables pour une prise en compte directe de la contrainte épipolaire. Vu ces opérations, le correspondant du pixel m_1 de l'image gauche rectifiée, est sur la même ligne dans l'image droite rectifiée (ce qui apparaît explicitement en **figure 1.1**) la réciproque est évidemment vraie.

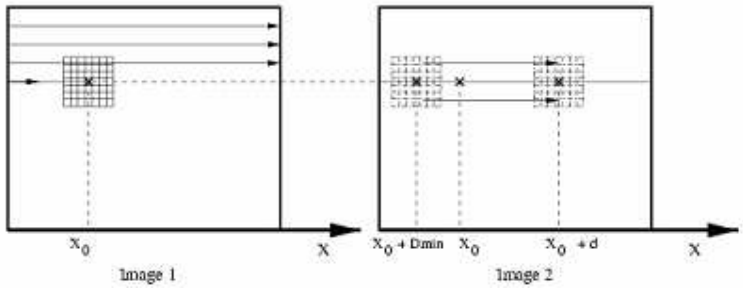

Figure 1.1 La mise en correspondance dans les images rectifiées

Les lignes seront donc traitées successivement ; pour un pixel en position x0 sur une ligne de l'image gauche, son correspondant sera recherché dans un intervalle de disparité [Dmin, Dmax] sur la même ligne de l'image droite. Sur des images rectifiées, on a généralement Dmax = 0 (point 3D à l'infini) et Dmin = - f (dist_min).

L'écart est maximal (en valeur absolue) pour une distance minimale des points appariés, distance dépendant de l'application et de la configuration géométrique du banc stéréo.

Pour savoir quel pixel dans cet intervalle est le correspondant de x0, on utilise un critère de ressemblance local: la corrélation sur les niveaux de gris dans un voisinage carré, centré sur les pixels analysés. Ce voisinage est appelé fenêtre de corrélation. La taille de cette fenêtre est un paramètre important de l'algorithme : cela va habituellement de 3 x 3 à 15 x 15.

Différents critères numériques de corrélation ont été proposés comme critère de ressemblance entre deux fenêtres de corrélation, une prise autour du pixel m'1 de l'image gauche, l'autre prise autour d'un pixel candidat pris sur la même ligne de l'image droite. Parmi ces critères on peut citer **[NAO06]**:

- ✓ SSD : Sum of Squared Differences, ou somme des différences quadratiques entre termes correspondants des deux fenêtres.
- ✓ SAD : Sum of Absolute Differences, ou somme de la valeur absolue des différences entre termes correspondants des deux fenêtres.
- ✓ ZSSD: Zero mean Sum of Squared Differences, identique à SSD sauf que, au préalable, pour chaque fenêtre de corrélation, la moyenne des termes est d'abord calculée et soustraite à chaque terme: la différence est donc faite entre les écarts à la moyenne des termes, et non entre les termes eux-mêmes. Ceci permet d'obtenir un critère invariant à des translations uniformes des luminances dans une des images.
- ✓ ZSAD: Zero mean Sum of Absolute Differences, identique à SAD, mais différence entre écarts à la moyenne. C'est donc un critère invariant aux variations uniformes de luminance dans une des images.
- ✓ NCC : Normalized Cross Correlation, le critère le plus utilisé, mais aussi le plus long à calculer. C'est la somme des produits entre les termes correspondants des fenêtres, normalisée par le produit des moyennes quadratiques calculées pour chacune des fenêtres.
- ✓ ZNCC : Zero mean Normalized Cross Correlation, identique à NCC mais corrélation croisée entre écarts à la moyenne.

Le critère ZNCC est le plus couramment utilisé. Il s'applique aux écarts à la moyenne dans chaque fenêtre, il est donc invariant aux variations uniformes de sensibilité entre les deux caméras, et par ailleurs, il est normalisé : le critère est compris entre 0 et 1, et il vaut 1 pour deux fenêtres identiques. Il est donc plus simple de définir un seuil de corrélation pour décider si deux fenêtres sont identiques ou différentes (par exemple, le score ZNCC doit être supérieur à 0.9 pour décider de l'appariement entre deux fenêtres).

Ces critères numériques, en particulier le critère ZNCC, sont longs à calculer. Les critères SAD ou ZSAD sont plus simples, mais, généralement, ils s'avèrent peu robustes.

Pour des applications temps réel, il a été proposé par Zabih et Woodfill **[ZAB94]**, un critère non numérique, plus qualitatif, appelé CENSUS. D'après la comparaison effectuée au LAAS par S. Gautama **[GAU99],** ce critère est plus robuste au bruit et plus efficace que les critères paramétriques sur des images peu texturées comme on en rencontre dans le milieu médical.

Au préalable, sur les images gauche et droite à comparer, une transformation est appliquée afin de remplacer la luminance en chaque pixel, par une chaîne de bits, dépendant de la comparaison de la valeur de ce pixel avec les autres pixels de la fenêtre de corrélation **[NAO06].**

Par exemple, pour la fenêtre présentée en **figure 1.2**, la valeur 67 de luminance pour le pixel central est remplacée par la chaîne montrée à droite. Les bits de cette chaîne

donnent les résultats des comparaisons de cette valeur (67) avec les valeurs de luminances des pixels de la fenêtre, parcourue ligne à ligne, de gauche à droite, et de haut en bas : le bit de comparaison vaut 1 si 67 est plus grand ou égal à la valeur du pixel traité, 0 sinon.

Par exemple, le premier bit est 0 car 120 > 67, le second 0 car 96 > 67, le troisième 1 car 65 < 67 ….

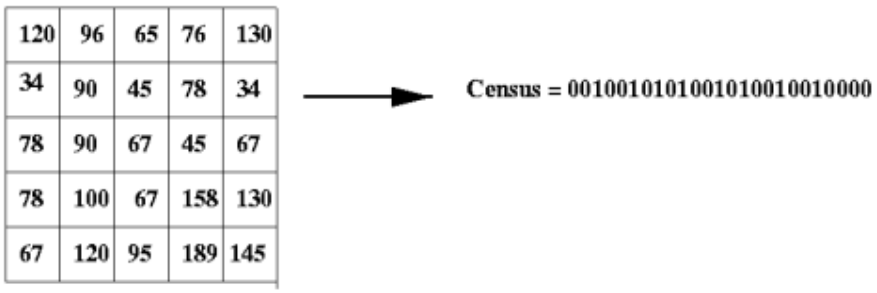

Figure 1.2 : Transformation Census sur un pixel avec une fenêtre 5 x 5

Une fois que ces transformations sont faites sur les deux images, le score utilisé pour comparer la chaîne associée à un pixel m'1 de l'image gauche, avec celles associées aux pixels de la même ligne de l'image droite, est la distance de Hamming entre chaînes de bits, c'est à dire le nombre de bits identiques entre deux chaînes, ou le nombre de bit égal à 1 dans le OU-exclusif entre deux chaînes.

Dans la réalisation pratique, différentes solutions peuvent être essayées **[NAO06]** :

- Calcul de la transformation CENSUS sur 8 bits seulement (fenêtre 3x3), mais utilisation d'un filtre de moyennage sur les images CENSUS pour prendre en compte en chaque pixel, un voisinage plus large ; elle est plus simple à programmer, car les images CENSUS restent des tableaux d'octets.
- Calcul de la transformation CENSUS sur N bits (par exemple 168 bits pour une fenêtre 13x13) et comparaison directe des images CENSUS sans filtrage préalable.

Pour une implémentation logicielle, cela pose le problème de gérer des images avec des pixels codés sur 5, 6, 7…. Octets, ce qui peut s'avérer assez peu performant en terme de temps calcul.

Pour trouver le correspondant du pixel x0 de l'image gauche, le critère est calculé pour tous les pixels candidats de l'image droite, appartenant à l'intervalle de disparité. Pour chaque association possible (x0, x0+d), on a donc un score de corrélation. L'ensemble des scores donne.

Le score ZNCC est compris entre 0 et 1 ; il est optimal (fenêtre de corrélation identique) quand il vaut 1. Le correspondant sera donc donné par le score maximum.

Plusieurs tests heuristiques permettent d'éviter les faux appariements :

- Le score maximum doit être supérieur à un seuil (ressemblance minimale exigée).
- L'écart avec le deuxième score maximal doit être significatif.
- Le pic de corrélation doit être assez étroit.

Si ces critères sont satisfaits, alors l'algorithme calcule une valeur de disparité sub-pixel, en faisant une simple interpolation parabolique entre le score maximum et les deux voisins. La sortie de cette étape est une image flottante de disparité qui contient pour chaque pixel de l'image gauche:

- 0 s'il n'a pas de correspondant.
- La valeur de disparité avec le correspondant sinon.

3. La détection d'obstacles par stéréovision

3.1 Introduction

La détection d'obstacles est une fonction classique en robotique mobile et se développe de plus en plus pour l'assistance au conducteur dans le domaine automobile. Dans le contexte des Systèmes de Transport Intelligents, la détection des obstacles routiers est une tâche essentielle. Différents capteurs embarqués peuvent être utilisés dans cette optique : radar, télémètre laser, vision (une ou plusieurs caméras). Les deux premières technologies restent onéreuses et présentent l'inconvénient de constituer une source de pollution électromagnétique active. La vision ne présente pas ces inconvénients mais souvent, les algorithmes de détection existants ne sont pas assez robustes, génériques ou rapides (en terme de temps de calcul) pour pouvoir être utilisés de façon efficace dans le contexte automobile, où tous les obstacles doivent être détectés en temps réel, et surtout, où il faut éviter les fausses alarmes.

3.2 L'approche de V-Disparité pour la détection des obstacles

Cette approche est basée sur la construction et l'analyse de l'image « v-disparité », qui fournit une bonne représentation géométrique de la scène routière **[LAB04]**.

3.2.1 Vue d'ensemble de l'algorithme

Une vue d'ensemble de l'algorithme est présentée sur la **Figure 1.3**. Une paire d'images stéréoscopiques est acquise. Une carte de disparité éparse est alors calculée, puis l'image « v-disparité » est construite. L'analyse de cette image permet l'extraction des surfaces globales dans la scène routière. Toutes les informations nécessaires à la détection des obstacles sont alors déduites (position des obstacles). **[LAB04]**

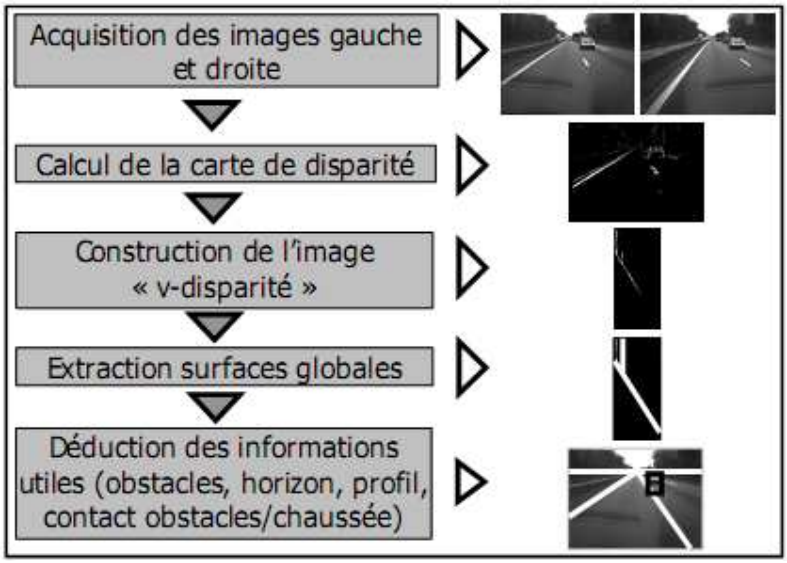

Figure 1.3 Vue d'ensemble de l'algorithme

3.2.2 Description de l'algorithme

La Figure 1.4 présente la modélisation du capteur stéréoscopique. Les deux plans images sont supposés parallèles et à la même hauteur au-dessus de la route. Cette configuration implique que la géométrie épipolaire est rectifiée (les lignes épipolaires correspondent aux lignes de balayage image). Les paramètres sur la figure 1.4 sont comme suit:

- b est la base stéréoscopique.
- h est la hauteur du capteur par rapport au sol.
- Θ est l'angle de tangage entre l'axe optique et l'horizontale.

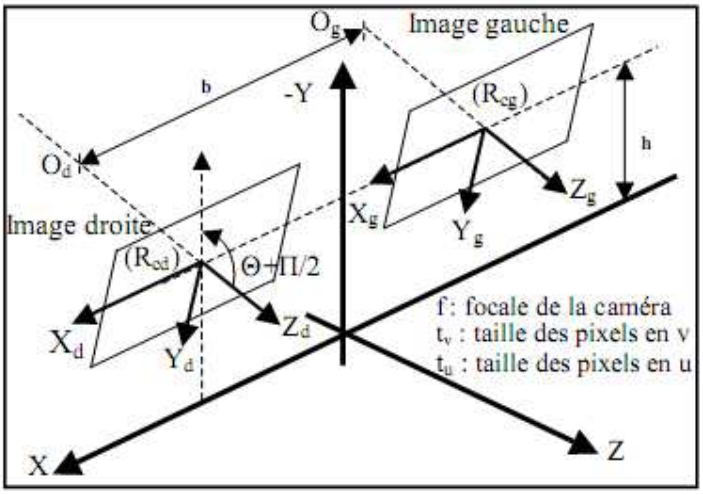

Figure 1.4 : Modélisation du capteur stéréoscopique

a) Construction de l'image « v-disparité »

Nous supposons qu'une carte de disparité a été calculée à partir de la paire d'images stéréoscopiques. Par exemple, une carte de disparité éparse, calculée en respectant la contrainte épipolaire. A ce stade de l'algorithme, on ne cherche pas à éviter les erreurs d'appariement mais à réduire le temps de calcul.

Une fois cette carte de disparité calculée, l'image « v-disparité » est construite en accumulant pour chaque ligne image tous les pixels de disparité connue.

Ainsi, pour la ligne image i, l'abscisse u_m du point M dans l'image « v-disparité » correspond à la disparité de ce point Δ_M et la valeur de son niveau de gris correspond au nombre de points de même disparité sur la ligne i. Le processus d'accumulation ainsi mis en œuvre est essentiel et confère à la méthode ses propriétés de robustesse. **[NAS09]**.

b) Analyse de l'image « v-disparité »

L'analyse de l'image « v-disparité » va permettre d'extraire des surfaces globales dans la scène routière, tels les plans de la famille de plans d'équation Z=aY+d. La route est caractérisée par un ensemble de plans obliques et les obstacles sont caractérisés par des plans verticaux. Ce type de plan se projette en une droite sur l'image « v-disparité », d'équation : **[LAB04]**

$$\Delta = \frac{\mathrm{b}}{\mathrm{ah}-\mathrm{d}}\big((\mathrm{v}-\mathrm{v}_0)(\mathrm{a}\cos\theta+\sin\theta)+\mathrm{a}(\mathrm{a}\sin\theta-\cos\theta)\big) \qquad (1)$$

Où Δ est la disparité, v l'ordonnée d'un pixel dans le repère image, v_0 l'ordonnée de la projection du centre optique, et a = f/tu ≈ f/tv (tu et tv tailles des pixels en u et v) . L'extraction de ces droites revient à la détection de surfaces globales et est réalisée par un procédé robuste comme une transformée de Hough.

c) Déduction des informations utiles

Une fois les surfaces globales détectées, il est aisé de déduire géométriquement diverses informations utiles pour la détection d'obstacles :

- La ligne de contact obstacle-chaussée est localisée à l'intersection entre la surface de la route et le plan obstacle et est ainsi déduite immédiatement de l'intersection des projections de ces surfaces dans l'image « v-disparité ». La ligne d'horizon est obtenue par des considérations similaires.
- La distance D d'un obstacle est donnée par : **[LAB04]**

$$D = \frac{\mathrm{b}\,(\mathrm{a}\cos\theta-(\mathrm{v}_\mathrm{r}-\mathrm{v}_0)\sin\theta)}{\Delta} \qquad (2)$$

Où v_r est l'ordonnée du contact obstacle-chaussée dans l'image.

- La hauteur de l'obstacle est déterminée par la longueur de la ligne dans l'image de la V-Disparité. Sa largeur pourra être déterminée par une étude similaire dans un autre plan dit de l'U-Disparité **[NAS09]**.

4. Conclusion

Dans ce chapitre nous avons présenté d'une manière générale le domaine de la stéréovision, nous avons commencé par citer les différents types de capteur d'image et leurs principes, puis nous avons donné un état de l'art sur les méthodes de mise en correspondance en se basant sur la transformée de Census pour l'implémentée sur FPGA. Enfin, nous avons conclu ce chapitre par la méthode de V-Disparité pour la détection des obstacles. Dans le prochain chapitre, nous présenterons la transformée de Hough pour la reconnaissance des formes.

La transformée de Hough

1. Etat de l'art

La transformée de Hough (TH) est un outil de détection de courbes paramétriques dans l'image, elle été proposée par P.V.C Hough dans un brevet déposé en 1960**[HOU62].**

Inaperçu pendant plusieurs années, cette dernière a été vulgarisée par les travaux de Rosenfield **[ROU69]**, Duda et Hart **[DUD72]** au début des années 70 et fait l'objet par la communauté scientifique depuis cette date à ce jour d'une particulière attention, Depuis les années 80, elle a quitté les laboratoires de recherche pour trouver des champs d'applications dans de nombreux domaines industriels **[OFF85, TZV91]** tels que la vision par ordinateur et le traitement des images. Elle est devenue une solution plus adaptée au problème de détection des lignes droites, cercle ou autre forme paramétrique dans l'image. Cependant, le calcul de la TH requiert de larges délais de traitement et beaucoup d'espace mémoire, alors plusieurs nouveaux algorithmes tel que la TH probabiliste, la TH aléatoire, la TH hiérarchique, la TH incrémentale, ont été proposés pour améliorer ce calcul, le rendre efficace et praticable pour son utilisation dans le traitement d'images en temps réel.

Pour illustrer le principe de base de la TH, on considère par exemple, le problème de détection d'un objet polyédrique dans une image. On doit avant toute détection, extraire les traits de ce dernier puis reconstruire son image. Donc on est amené à extraire des lignes droites ou autrement dit les ensembles des points colinéaires. La méthode robuste qui peut faire ce traitement doit tester la présence de droites formées par toutes les paires de points de l'image, ce qui est extrêmement inefficace, vu que pour tester n points de l'image deux par deux, on arriverait à un nombre exagéré d'itérations au moins supérieur à n^2.

Dans une image 512x512 cette méthode devient prohibitive **[BOU06]**. La TH résoud ce problème car elle transforme les lignes de plan 2D de l'image en points dans le plans des paramètres qui définit ces lignes, par conséquent, elle convertit le problème de détection de ligne en un autre plus simple celui de la détection des points d'intersection. Le principe est de prendre toutes les paires de pixels appartenant aux contours de l'image et construire un histogramme à deux dimensions appelé aussi le « tableau accumulateur » qui servira à enregistrer les droites ainsi trouvées de chaque paire de pixels tirée. Tout point de l'image faisant parti d'une de ces droites se voit incrémenter d'un crédit dans l'histogramme.

Tout en gardant la même idée principale du calcul de la TH citée ci-dessus ; ils existent plusieurs façon de traiter les points de l'image, ce qui nous donne diverses techniques de la TH. Celles-ci sont énumérées comme suit :

a) Transformée de Hough probabiliste

Contrairement à la transformée de Hough standard qui applique la TH pour tous les pixels d'une image contours, la transformée de Hough probabiliste **[BOU06]** affirme qu'il suffit de calculer la TH seulement pour une portion α de pixel de cette image ($0<\alpha<100$). Ces pixel sont choisis aléatoirement à partir de la densité de probabilité uniforme définie sur l'image. Kitter et al **[KIT88]** préconisent d'utiliser une valeur α comprise entre 10% et 20%, cette variation dépend de l'application.

b) Transformée de Hough aléatoire

La TH aléatoire est présentée dans **[KIT88]** et les explications peuvent être aussi trouvées dans **[AIR94, MUR95]**. Brièvement, la TH aléatoire, utilise une autre technique pour générer des valeurs dans le tableau accumulateur définit sur le plan de paramètres. Dans le cas de détection d'une ligne droite dans la TH standard, un pixel de l'image correspond à une courbe dans le plan de paramètres, celui-ci est discrétisé et enregistré dans le tableau accumulateur, par contre dans la TH aléatoire, une paire de pixels est choisie aléatoirement et les paramètres de la ligne unique qui passe par ces pixels sont calculés.

Cette ligne est enregistrée comme la seule entrée dans le tableau accumulateur et les pixels de cette dernière sont ensuite enlevés, et laissent une image simple à analyser. De cette façon, les entrées sont accumulées dans l'espace des paramètres. Cet algorithme est ensuite répété pour détecter les lignes suivantes un nombre de fois dans le temps, ou le nombre des itérations est beaucoup moins que le nombre de paire de pixels dans l'image. L'algorithme s'arrête quand aucune ligne n'est détectée pour ce nombre d'itérations.

c) Transformée de Hough hiérarchique

La TH hiérarchique combine une structure pyramidale avec la transformée de Hough (**figure 2.1**). L'image est organisée en grille de sous images et la TH est appliquée sur chacune d'elle. Typiquement, chacun des ces sous images va contenir au plus deux lignes. Ces résultats sont propagés vers le haut à travers la pyramide. Pour chaque nœud du prochain niveau, on prend le chevauchement à l'aide d'un masque 4x4 du niveau précédent. Les lignes de ces dernières sont fusionnées en utilisant l'algorithme de la TH. Ces lignes sont propagées au prochain

niveau hiérarchique. Cet algorithme est répété jusqu'au niveau le plus haut de la pyramide, où on trouve un seul nœud. Ce nœud représente alors l'image entière. Les détails sont donnés dans **[LOT94].**

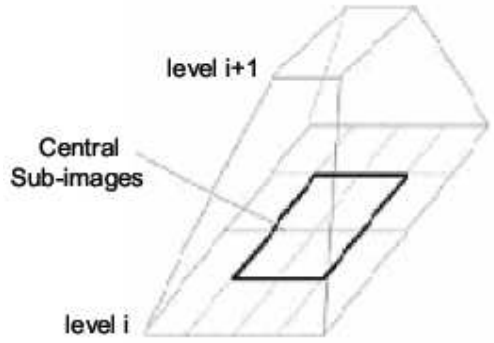

Figure 2.1 Structure pyramidale de la transformée de Hough hiérarchique

d) Transformée de Hough incrémental

Pour appliquer la TH dans les tâches de traitement d'image en temps réels, son calcul doit être le plus court possible. Habituellement la TH dans le cas particulier des droites, est définit par l'équation :$f(x, y, \rho, \theta) = \rho - xcos\theta - ysin\theta = 0$. Donc, son calcul réclame l'utilisation des formules trigonométrique et des multiplications, ce que nécessite un temps de calcul énorme. Pour remédier à ce problème, on utilise la transformée de Hough incrémental. Cette dernière utilise une autre expression de la TH, définie par des fonctions d'additions et des décalages **[ACH04].**

2. principe de la transformée de Hough

Une droite est décrite dans le plan cartésien(x, y) par l'expression suivante :

$f(y, x, a, b) = y - ax - b = 0$ (1)

Sachant que $\boldsymbol{a}$ est la pente et $\boldsymbol{b}$ l'ordonnée à l'origine des abscisses.

Etant donné un ensemble de conteurs d'objets représentés par un ensemble de points discrets M_i, nous cherchons à déterminer si un ou plusieurs sous ensembles de points M_i font partie d'une courbe dont les paramètres a et b restent à définir. Si nous cherchons à tester les n points Mi deux par deux, nous arriverons à un nombre exagéré d'itérations au moins supérieur à n^2.

Hough puis Rosenfeld **[HOU62, ROS69]** ont proposé une méthode pour détecter les droites à l'aide des points du plan (x, y). Son principe est de calculer pour chaque point Mi de

coordonnées (x_i, y_i), du contour d'un objet, l'ensemble des paramètres a qui vérifient l'équation $f(x_i, y_i, a, b) = 0$ avec b fixé.

Pour chaque point $M_i(x_i, y_i)$, de l'image, il y a un ensemble de valeurs possibles pour les paramètres $a\ et\ b$. cet ensemble forme une droite d'équation $b = -ax + y$

Dans l'espace des paramètres(a, b), qui se coupent au point N de coordonnées(a', b'). De cette façon tous les points qui appartiennent à la même droite forment des droites dans le plan des paramètres (a, b) qui se coupent au même point. Ce concept est illustré dans les **figures 2.2.a** et **2.2.b**.

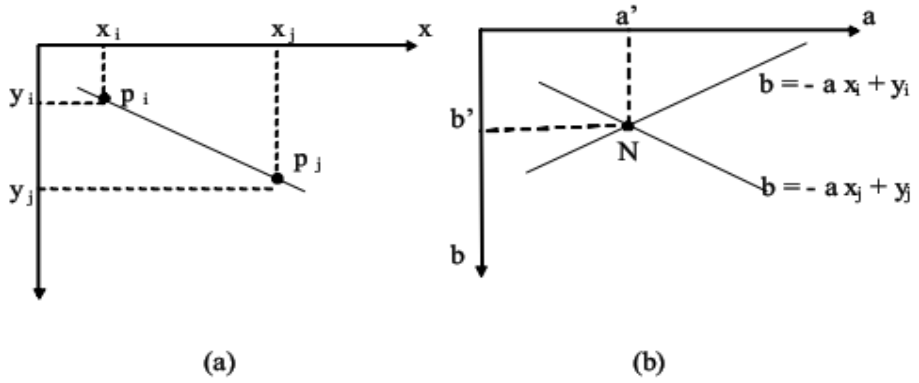

Figure 2.2. Transformée de Hough

a. Plan cartésien (x,y)

b. Plan des paramètres (a, b)

Le traitement Hough consiste en une quantification du plan des paramètres en cellules accumulatrices sur la figure 2.3 ou $(a_{min}, a_{max})\ et\ (b_{min}, b_{max})$ sont les valeurs limites de l'intervalle de la pente a et de l'ordonnée à l'origine des abscisses b.

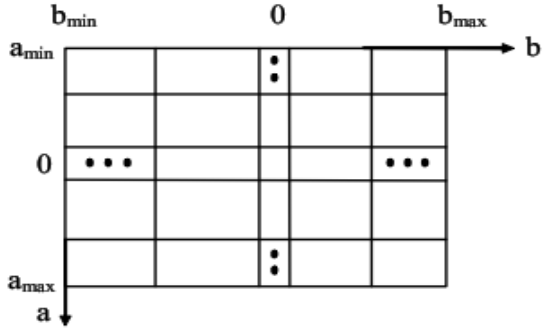

Figure 2.3. Quantification du plan des paramètres (ab)

Chaque cellule de coordonnées **(i, j)** a une valeur accumulée **A (i, j)** et correspond à la cellule de coordonnées $(\boldsymbol{a_i}, \boldsymbol{b_j})$ dans le plan des paramètres $(\boldsymbol{a}, \boldsymbol{b})$. initialement ces cellules sont mise à zéro. Pour chaque point de l'image de coordonnées$(\boldsymbol{x_k}, \boldsymbol{y_k})$, on calcule pour chaque valeur de $\boldsymbol{a}$ quantifié à la valeur $\boldsymbol{a_p}$ sur l'axe des $\boldsymbol{a}$, son correspondant $\boldsymbol{b}$ en utilisant l'équation suivante : $\boldsymbol{b = -ax_k + y_k}$. La valeur résultante, résultat $\boldsymbol{b}$ est arrondi à la valeur la plus proche de $\boldsymbol{b}$ quantifié $\boldsymbol{b_q}$ sur l'axe des $\boldsymbol{b}$. Si on obtient une valeur $\boldsymbol{b_q}$ suite à $\boldsymbol{a_p}$ choisie, on incrémente la valeur de la cellule correspondante : $A(p, q) = A(p, q) + 1$.

A la fin de cette procédure, la valeur n de $A(i, j)$ dans une cellule (i, j)correspond à $\boldsymbol{n}$ points dans le plan (x, y) qui vérifient l'équation $y = a_i x + b_j$, donc il existe $\boldsymbol{n}$ points qui appartient à la droite de pente $\boldsymbol{a_i}$ et de l'ordonnés à l'origine des abscisses$\boldsymbol{b_j}$.

L'inconvénient majeur de cette procédure réside dans son incapacité de détecter les droites verticales. Pour remédier à ce problème, un paramétrage polaire (p, θ) est plus satisfaisant. Ce paramétrage est illustré dans la **figure 2.4.**

Une droite est alors définie par l'équation suivante :

$$\boldsymbol{f(x, y, p, \theta) = p - xcos\theta - ysin\theta = 0} \qquad (\mathbf{2})$$

Avec $p\ la\ distance\ perpendiculaire\ à\ la$ droite de l'origine du plan $(x, y)\ et\ \theta\ l'angle\ entre$ cette distance et l'axe des x.

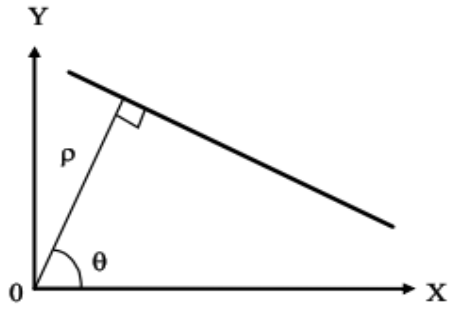

Figure 2.4. Paramétrage polaire d'une droite

L'utilisation de cette représentation dans la construction de tableau accumulateur est identique à celle développée précédemment (représentation ab sur la **figure 2.5**).

On précisera que le choix de quantification de l'espace des paramètres (p, θ) doit porter sur les trois objectifs essentiels suivants :

1. Garantir une précision de détection aussi bonne que possible.
2. Diminuer la mémoire nécessaire au stockage des accumulateurs.
3. Accélérer les calculs.

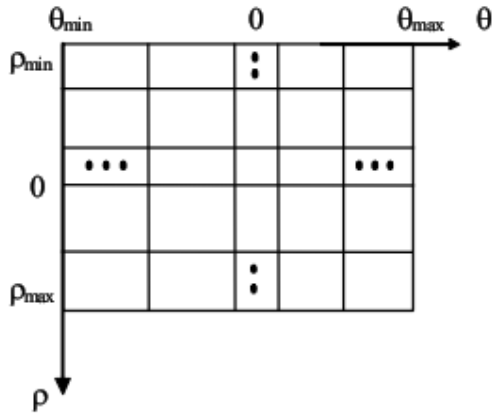

Figure 2.5. Quantification du plan des paramètres (p, θ)

Chaque point M_i de coordonnées(x_i, y_i) d'une droite se transforme dans le plan des paramètres (ab)en une sinusoïde d'équation : $p = x_i cos\theta + y_i sin\theta$.

Donc, une droite sera représentée par un ensemble de sinusoïdes qui se coupent en un seul point de coordonnées polaires (p_0, θ_0) caractéristique de cette droite dans le plan des paramètres (**figure 2.6-a** et **2.6-b**).

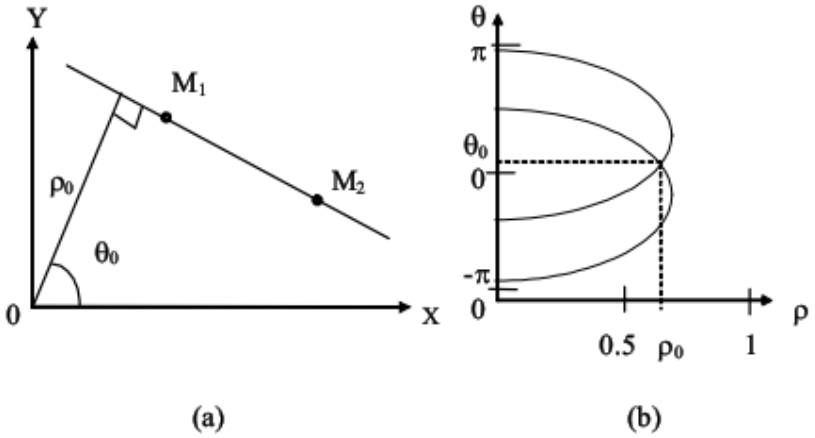

Figure 2.6. Transformée de Hough

(a) Plan cartésien (x, y)

(b) Plan des paramètres (a, b)

3. Dimension des paramètres θ et p

3.1 Champ de dimension de θ

Le champ de dimension de θ est [0,2π]. Pour un angle θ appartenant à cet intervalle, toutes les droites s'expriment avec un p positif. Nous remarquons que les droites dont θ appartenant à l'intervalle [π, 2π] peuvent être vues comme des droites à p négatif. Donc l'intervalle de θ peut être réduit de moitié, c'est-à-dire le champ de θ sera [0,π] et pour chaque valeur θ appartenant à cet intervalle toutes les droites s'expriment avec un p pas strictement positif. En effet si on considère la droite D2 avec les paramètres polaires (θ_2, p_2) avec : $\theta_2 = \pi + \theta_1$

$p_2 = x\cos\theta_2 + y\sin\theta_2$

$p_2 = x\cos(\pi + \theta_1) + y\sin(\pi + \theta_1)$

$p_2 = -(x\cos\theta_1 + y\sin\theta_1)$

$p_2 = -p_1$

La droite D2 de la figure 2.7 est vue comme la droite D1 mais avec un p opposé ($p_2 = -p_1$).

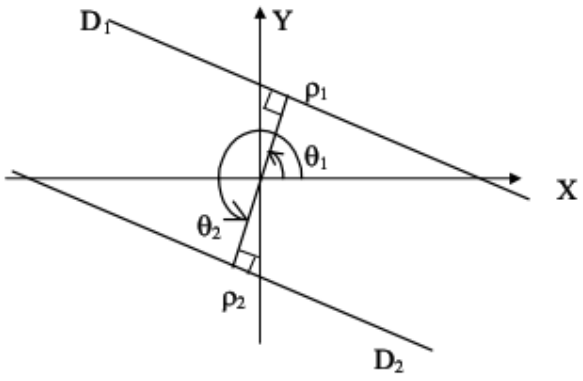

Figure 2.7. Paramètres polaires de deux droites opposées

3.2. Champ de dimension de p

Le champ de dimension de p est défini selon la taille de l'image. Pour une image carrée NxN. A partir de la figure 2.8, on trouve que le champ de dimension de p est [0, N$\sqrt{2}$].

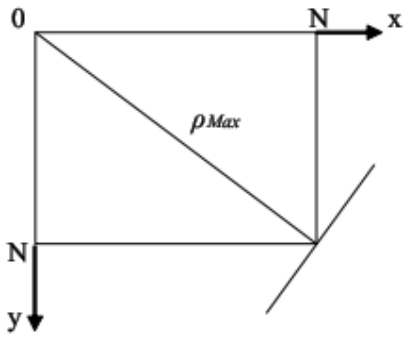

Figure 2.8. Le champ de la dimension de p.

Si nous déplaçons l'origine O du repère du plan (x, y) au centre de l'image, puis nous examinons les valeurs $p_{min}\ et\ p_{max}$ de la dimension de p.

p_{max} est donnée par la plus grande distance de la droite par rapport à l'origine O. Cette droite est représentée par la droite D2 dans la **figure 2.9** et $p_{max}={}^{Max}/_{\sqrt{2}}$.

p_{min} est donnée par la plus petite distance négative de la droite par rapport à l'origine O. Cette droite est représentée par la droite D1 dans la **figure 2.9** et $p_{min}={}^{-Max}/_{\sqrt{2}}$.

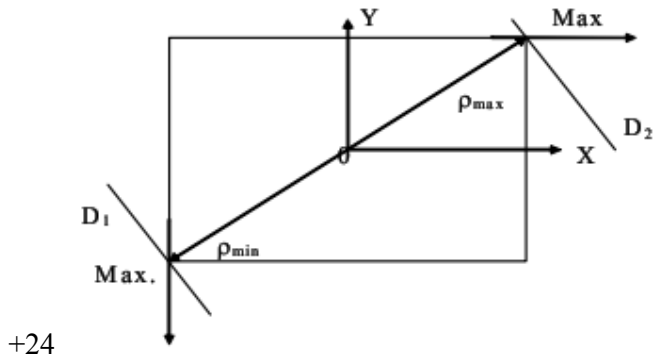

Figure 2.9 Le champ de dimension de p

3.3. Propriétés de la transformée de Hough

- Un point du plan cartésien correspond à une sinusoïde dans le plan des paramètres.
- Un point de plan des paramètres correspond à une droite dans le plan cartésien.
- Les points appartenant à la même droite dans le plan cartésien correspondent au point d'intersection des courbes du plan des paramètres.

- Les points appartenant à la même courbe du plan des paramètres correspondent à des droites du même point du plan cartésien.

4. Utilisation de la transformée de Hough

L'extraction des lignes droites utilisant le tableau accumulateur de Hough est une tâche qui nécessite la prise de certaines précautions car les n points, d'une cellule du tableau accumulateur, n'indique pas s'ils sont prochés ou dispersés. Ceci induit le problème d'extraction de fausses droites ou droites insignifiantes. Ce problème est accentué lorsqu'on a une image trop éclairée (création de zones d'ombre) et une image qui possède un nuage de points (une image trop bruitée).

Pour résoudre ce problème, on élimine d'abords tous les segments de droite ayant un nombre de points réduit dans le tableau accumulateur, ensuite on scrute les cellules accumulatrices de ce dernier en cherchant les droites significatives **[ATI92, MAR97].**

Ces droites sont des droites réelles qui existent dans l'image telles que les arrêtes d'un objet quelconque dans cette image. Une droite significative se présente par un pic (sommet d'une montagne), lorsqu'on représente le tableau accumulateur en 3D sur la **figure 2.10.** Une droite significative doit avoir les caractéristiques suivantes :

- Sa longueur ne doit pas être trop petite devant un seuil donné (seuilH).
- La longueur des segments de droite qui forment cette droite ne doit pas être inferieure à un seuil donné (seuilC), sinon, ils seront éliminés.
- Les points qui forment ces segments de droite ne doivent pas être trop espacés, donc la distance séparant deux points ne doit pas être supérieure à un troisième seuil donné (seuilD).

Donc pour avoir une bonne détection des droites significatives, il faut faire un compromis entre les trois seuils seuilH, seuilC et seuilD suivant le domaine d'application utilisée.

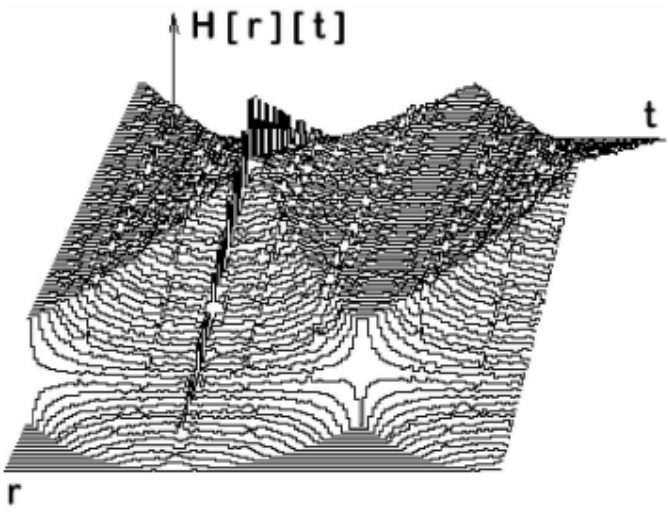

Figure 2.10 Le plan de Hough en relatif

Nous remarquons aussi qu'il y a présence des vallées et des montagnes avec leurs pics. Les pics des montagnes correspondent aux droites réelles qui existent dans l'image, par contre les vallées ne correspondent en réalité à rien car elles sont les conséquences du calcul de la transformée de Hough.

5. Implémentation de la transformée de Hough

La transformée de Hough est un outil puissant dans l'analyse des formes est utilisée aussi pour l'extraction des traits globaux des formes dans l'image. Elle donne de bons résultats, même en présence de bruits dans l'image. La transformée de Hough (TH) opère sur des données binaires des points contours de l'image. Chaque point contour de coordonnées (x_i, y_i) de l'image est transformé en une sinusoïde dans le plan des paramètres. Elle contribue dans ce plan par l'incrémentation de la cellule (p_i, θ_i) du tableau accumulateur créé par cette transformation où θ_i est une valeur discrète suivant la résolution choisie de θ dans l'intervalle [0, π]. Initialement les cellules du tableau accumulateur sont mises a zéro **[TZV91, DOL94].** Les points contours colinéaires de l'image produisent des sinusoïdes dans les plans des paramètres qui se croisent en un point commun (p , θ), avec p la distance normale à la droite qui porte ces points colinéaires de l'origine de l'image et θ l'angle entre cette distance avec l'axe des X. Nous affectons dans la cellule (p , θ) du tableau accumulateur le nombre de croisement des courbes au point d'intersection (p , θ) dans le plan des paramètres. La quantification du plan des paramètres p, θ revient à

quantifier l'intervalle $0 \leq \theta < \pi$ pour la dimension de θ et l'intervalle $-R \leq p < R$ pour la dimension de p, avec R la moitié de la diagonale de l'image. De plus, si nous prenons :

$p_k \; et \; \theta_k$ pour pas de quantification des dimensions de p et θ.

$n_p \; et \; n_\theta$ le nombre de valeur discrètes dans les intervalles de p et θ.

Les valeurs de p et θ discrétisées s'écrivent comme suit :

$\theta = t * \theta_k \quad 0 \leq t \leq n_\theta$ (3)

$p = -R + r * p_k \quad 0 \leq r \leq n_p$ (4)

Avec $n_\theta = \pi/\theta_k$ et $n_p = 2R/p_k$ (5)

Pour chaque point contour de coordonnées (x, y) de l'image, nous calculons pour chaque valeur discrète de θ la valeur discréte de p suivant l'équation suivante :

$p = x * cos\theta + y * sin\theta$,

Ensuite, nous incrémentons la cellule correspondante dans le tableau accumulateur. Nous faisons la même chose pour tous les points contours de l'image.

L'organigramme ci-dessous de la **figure 2.11** décrit ce calcul.

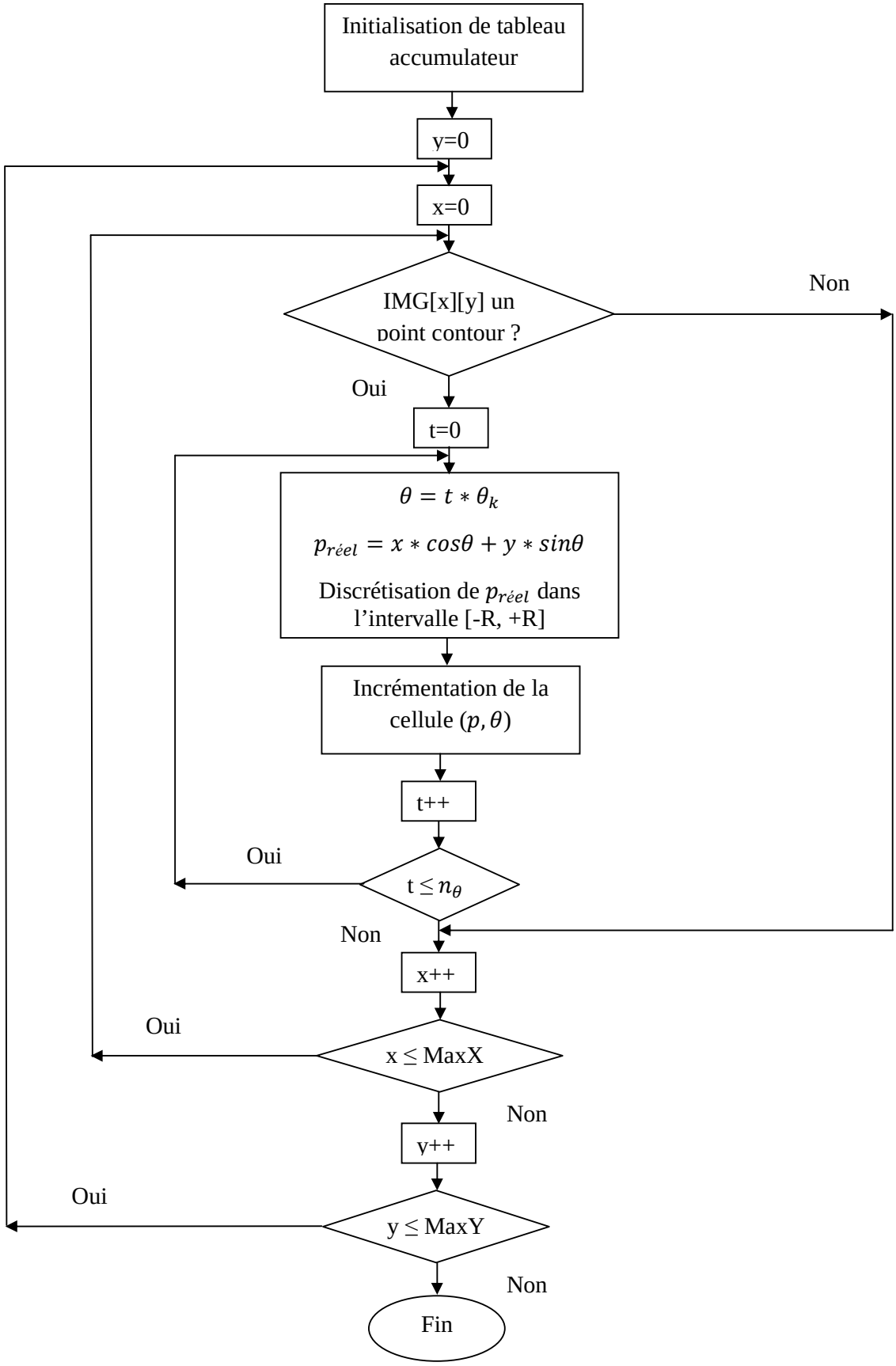

Figure 2.11 Organigramme de la TH

A la fin de ce calcul, une valeur M se trouvant dans une cellule quelconque (p, θ) du tableau accumulateur indique que M points de limage appartiennent à une même droite dont les paramètres polaires sont $p \; et \; \theta$. Mais, nous ne pouvons pas savoir si ces M points de l'image sont rapprochés ou dispersés. C'est le problème d'existence de fausses droites ou droites insignifiantes **(figure 2.12).**

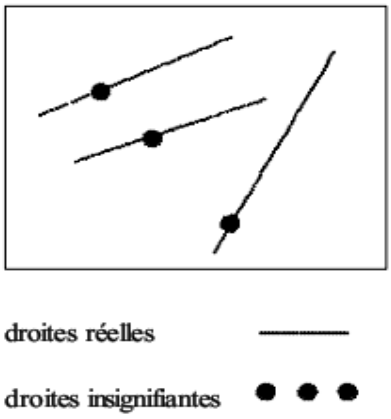

Figure 2.12. Droites réelles et insignifiantes

Donc pour avoir une bonne détection des droites significatives, il faut faire un compromis entre les trois seuils seuilH, seuilD et seuilC suivant le domaine d'application **[BOU06]**.

6. Conclusion

La transformée de Hough est une méthode de vote très robuste. La version originale de la méthode proposée par Hough a été modifiée par **[DUD72].** Depuis plusieurs variantes ont été proposées **[KIT88].** Cette approche repose sur une discrétisation de l'espace des paramètres. On obtient alors des hypercubes dans l'espace d'état auquel sont associés des accumulateurs. Pour un jeu de données de taille minimale, les paramètres recherchés sont estimés et l'accumulateur correspondant de l'hypercube est incrémenté. Ce processus est itéré jusqu'a considérer toute les combinaisons possibles des données à disposition. L'accumulateur ayant la valeur la plus importante correspond alors à la meilleure estimation des paramètres.

L'implémentation de la transformée de Hough sur un système informatique a l'avantage de présenter un large éventail de tâches à exécuter en parallèle. Ce système informatique peut exécuter d'autres algorithmes en plus de la transformée de Hough et être capable de donner une performance aux calculs antérieurs tels qu'une segmentation ou aussi une détection de contours.
Par conséquent, dans le chapitre qui va suivre, nous présentons les outils de développement tel que la carte RC203E et les outils logiciels utilise.

Les outils de développement

1 Introduction

Pour la plupart des algorithmes de traitement d'images, le nombre d'opérations à effectuer dépend généralement de la taille des images à traiter. Cette grande masse d'opérations engendre l'accroissement rapide du temps d'exécution d'où la nécessité de circuits électroniques rapides pouvant accélérer le traitement en se basant sur des architectures parallèles et pipelines.

Dans ce chapitre, nous présenterons les différents circuits numériques (ASICs, DSPs, FPGAs) dédiés au traitement d'images. Nos nous intéresserons à l'étude des circuits FPGAs, et leurs avantages dans le traitement des images. Après, nous détaillerons la plateforme de développement, la carte RC203E de Celoxica qui est à base du circuit FPGA Virtex-II. A la fin, nous présenterons les plateformes logiciels de développement, particulièrement la plateforme DK et le langage Handel-C.

2. Les différentes architectures des circuits numériques

De nombreuses familles de circuits programmables et reprogrammables sont apparues depuis les années 70 avec des noms très divers suivant les constructeurs**.** Ici, nous citerons les circuits les plus répandus dans le domaine du traitement d'images en temps réel.

2.1. Les ASICs (Application Specific Integrated Circuit)

Par définition, les circuits ASICs regroupent tous les circuits dont la fonction peut être ***personnalisée*** d'une manière ou d'une autre en vue d'une application spécifique, par opposition aux circuits standards dont la fonction est définie et parfaitement décrite dans le catalogue des composants. **[DJE05]**

Les ASICs peuvent être classés en plusieurs catégories selon leur niveau d'intégration, en fait un ASIC est défini par sa structure de base (réseau programmable, cellule de base, matrice, etc.). Sous le terme ASIC deux familles sont regroupées, les semi personnalisés, avec des réseaux prédéfinis et les personnalisés qu'on optimise pour créer son propre composant.

D'une manière générale l'utilisation d'un ASIC conduit à de nombreux avantages provenant essentiellement de la réduction de la taille des systèmes. Il en ressort, une réduction du nombre des composants sur le circuit imprimé. La consommation et l'encombrement s'en trouvent considérablement réduits. ·

Le concept ASIC par définition assure une optimisation maximale du circuit à réaliser. Nous disposons alors d'un circuit intégré correspondant réellement à nos propres besoins.

La personnalisation du circuit donne une confidentialité au concepteur et une protection industrielle.

Enfin, ce type de composant augmente la complexité du circuit, sa vitesse de fonctionnement et sa fiabilité.

Dans l'approche des circuits du type ASIC, l'inconvénient majeur réside dans le fait du *passage obligatoire chez le fondeur* ce qui implique des frais et un temps de développement élevés du circuit.

Les ASICs sont généralement plus convenables pour les productions en série de conceptions déjà vérifiées et non pour les prototypes.

2.2. Processeurs de traitement du signal(DSP)

Les DSPs (Digital Signal Processors) sont des processeurs dédiés au traitement du signal et des images, en particulier pour les traitements de type moyen niveau et haut niveau. Ces processeurs sont relativement semblables aux microprocesseurs. Leur particularité essentielle est qu'ils sont conçus pour effectuer des calculs en temps réel et intègrent donc de nombreux opérateurs. Ils permettent également un accès rapide aux données par des adressages particuliers. Leur jeu d'instructions est souvent plus réduit que celui d'un processeur traditionnel. Les DSPs peuvent être programmés soit en assembleur, soit en C en utilisant un compilateur dédié. Ces processeur ont la possibilité de réaliser plusieurs instructions en parallèle et possèdent des opérateurs de calculs arithmétiques flottants très performants **[KOB09]**.

2.3. Les FPGA"Field Programmable Gate Arrays" ou "réseaux des blocs programmables"

Les circuits FPGAs sont des composants entièrement reconfigurables ce qui permet de les reprogrammer à volonté afin d'accélérer notablement certaines phases de calculs. Les circuits FPGA sont constitués d'une matrice de blocs logiques programmables entourés de blocs d'entrée/sortie programmables. L'ensemble est relié par un réseau d'interconnexions programmable.

Les FPGAs ne sont pas optimisés pour une application bien déterminée, par conséquent ils consomment plus d'énergie que les ASICs. Par contre ils sont beaucoup plus simples à programmer et à reprogrammer, ce qui raccourcit les cycles de conception et permet de suivre l'évolution de l'application pour laquelle, ils ont été conçus.

Les FPGAs sont plus convenables pour les prototypes et pour les productions en série limitées qui ne sont pas de la qualité des ASICs. **[DJE05]**

Les circuits FPGA possèdent une structure matricielle de deux types de blocs (ou cellules)**(figure 3.1)**. Des blocs d'entrées/sorties(IOB) et des blocs logiques programmables(CLB). Le passage d'un bloc logique à un autre se fait par un routage programmable. Certains circuits FPGA intègrent également des mémoires RAM, des multiplieurs et même des noyaux de processeurs**. [MEC06]**

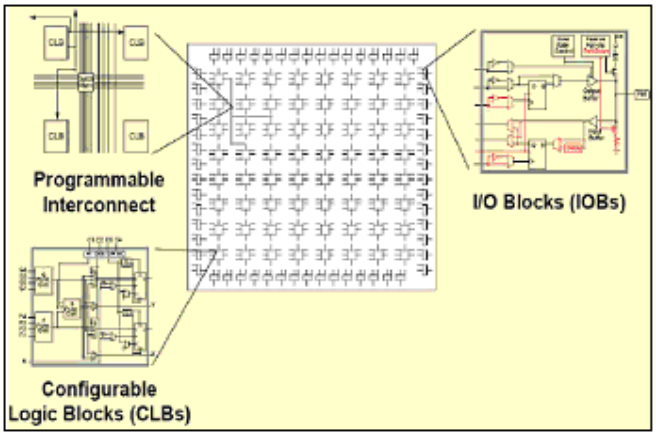

Figure 3.1 : Architecture interne du FPGA

2.3.1 CLB (Configurable Logic Bloc)

- C'est l'unité fondamentale du bloc logique qui fournit des éléments utilitaires pour la logique combinatoire et la logique synchrone, y compris les éléments du stockage de base : buffers à 3 états à associer avec chaque CLB. Les CLBs incluent quatre parties identiques appelées SLICE, **Figure. 3.2 [MER07]** et deux buffers à 3 états.
- Chaque SILICE est équivalente et contient :
  - Deux éléments du stockage.

- Des portes de la logique arithmétique.
- Grands multiplexeurs.
- Une large fonctionnalité.
- Une logique de retenue rapide.
- Une chaîne de cascade Horizontale (porte OU).

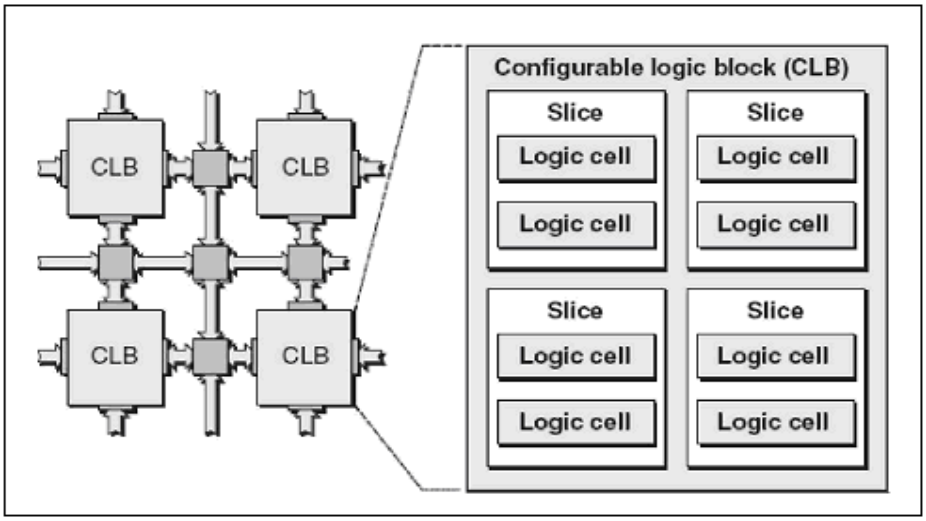

Figure 3.2 : Bloc CLB

Les blocs logiques configurables sont les éléments déterminants des performances du circuit FPGA (**figure 3.2**). Chaque CLB est un bloc de logique combinatoire composé de générateurs de fonctions à quatre entrées (LUT) et d'un bloc de mémorisation/synchronisation composé de bascules D.

Quatre autres entrées permettent d'effectuer les connexions internes entre les différents éléments du CLB.

La LUT (Look Up Table) est un élément qui dispose de quatre entrées, il existe donc $2^4 = 16$ combinaisons différentes de ces entrées. L'idée consiste à mémoriser la sortie correspondant à chaque combinaison d'entrée dans une petite table de 16 bits, la LUT devient ainsi un petit bloc générateur de fonctions. **La figure 3.3** montre le schéma simplifié d'un CLB de la famille XC4000 de Xilinx.

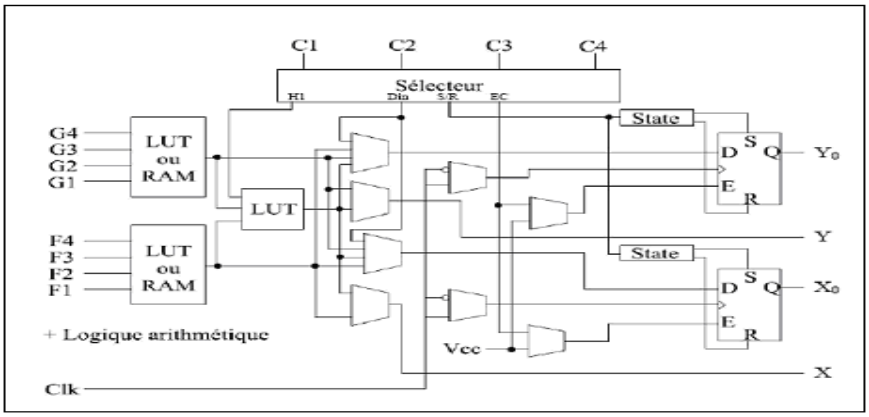

Figure 3.3 : Schéma d'une cellule logique(CLB) (XC4000 de Xilinx)

2.3.2 IOB (Input Output Bloc)

Ils constituent l'interface entre les bornes du circuit et les CLB. Le dispositif de routage Versa Ring offre les ressources nécessaires à l'interconnexion des CLB aux IOB. Nous pouvons ainsi modifier le système implanté sur le FPGA sans interférer avec l'attribution des bornes. Cette caractéristique s'avère importante si nous souhaitons développer des nouvelles versions d'un produit tout en conservant la compatibilité au niveau du boîtier.

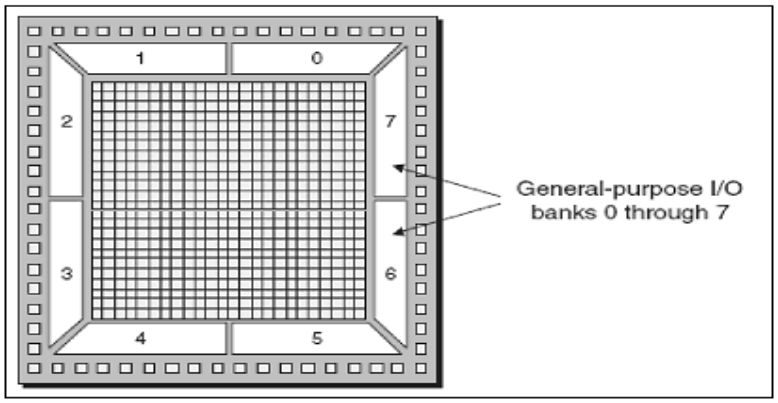

Figure 3.4 : Exemple de IOB

Ils sont présents sur toute la périphérie du circuit FPGA **(figure 3.4)**. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (état haute impédance). **La figure 3.5** présente la structure de ces blocs.

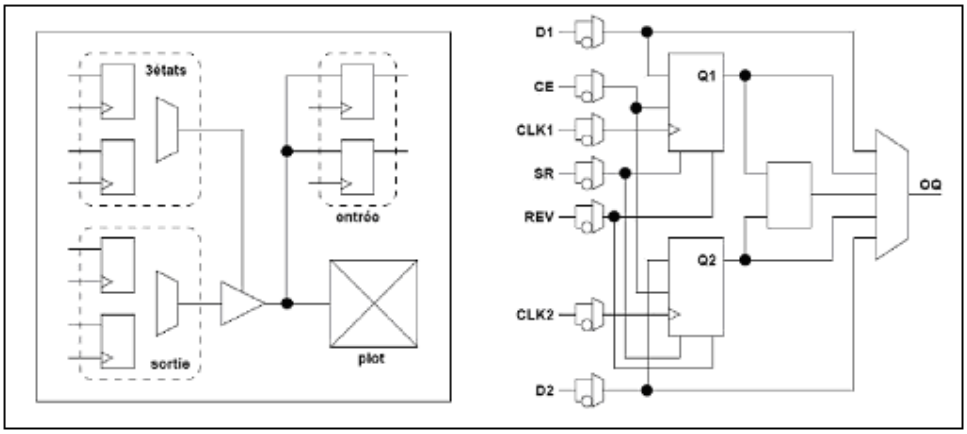

Figure 3.5 : Schéma d'un bloc d'entrée/sortie (IOB)

3. Avantages des FPGAs

- En général, le composant FPGA réalise un bon compromis entre la flexibilité et la performance par rapport aux processeurs classiques et ASICs **[SAK05].**

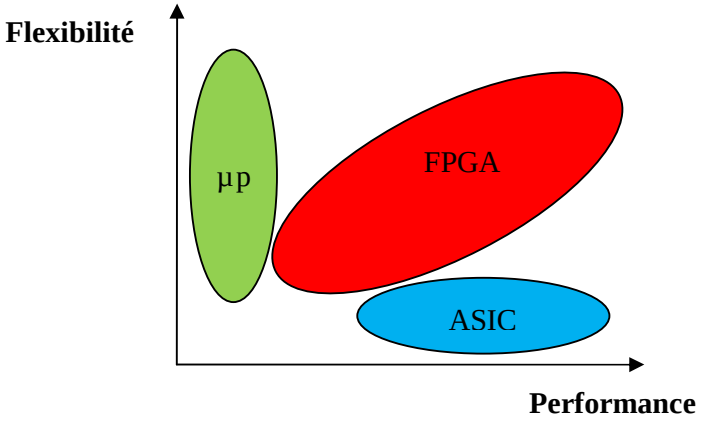

Figure 3.6: FPGA réalise le compromis Flexibilité/Performance

- les circuits programmables fournissent une puissance de calcul importante. Des algorithmes, traditionnellement considérés comme une séquence d'instructions, peuvent être analysés pour déterminer la possibilité d'un traitement parallèle. La possibilité de reconfiguration sur ces circuits permet une minimisation et une adaptation spécifique à chaque algorithme **[GHO03].**

- Le traitement des images en temps réel conduit à employer des circuits électroniques rapides capables de manipuler les grandes quantités d'informations générées par la source vidéo. Pour cela, les circuits logiques programmables sont particulièrement bien adaptés. Des circuits intégrés spécifiques(ASIC) pour des traitements particuliers, ne permettent pas de modifier la fonction de traitement une fois le circuit réalisé **[GHO03].**
- Un faible coût pour une production à faible unité, ou pour le développement de prototypes car ces circuits sont indéfiniment reprogrammables.
- Un temps de fabrication du circuit négligeable par rapport à un circuit ASIC développé chez un fondeur **[GHO03].**

4. Applications des FPGAs

- Le prototypage rapide : un FPGA permet de transférer une nouvelle application sur silicium en quelques heures, alors que la réalisation des circuits intégrés (ASIC) nécessite des semaines voir des mois. Ceci présente l'avantage de pouvoir évaluer plusieurs options d'architecture de réalisation pour une application donnée.
- L'émulation du matériel : ceci est particulièrement efficace pour des applications pour lesquelles la simulation logique s'avère inutilisable à cause d'événements survenant en temps réel. Un FPGA peut servir comme circuit de prototypage bon marché dans la mesure où le temps de conception est une tâche critique.
- L'accélération de fonctions : un FPGA peut fournir une aide précieuse pour le développement de différentes fonctions sur un composant reconfigurable. Par exemple, afin de satisfaire au mieux les besoins de communication entre plusieurs processeurs parallèles, les composants sont configurés selon l'interface à utiliser **[GHO03]**

5. Programmation et configuration des circuits FPGAs

La configuration est le processus de charger des données spécifiques à une conception dans un ou plusieurs FPGAs pour définir l'opération fonctionnelle des blocs internes ainsi que leur interconnexion. Le temps de configuration dépend du mode de configuration sélectionnée.

Dans la littérature on peut distinguer 4 modes de configuration et programmation des circuits FPGAs comme le montre **la figure 3.7 [DAO06].**

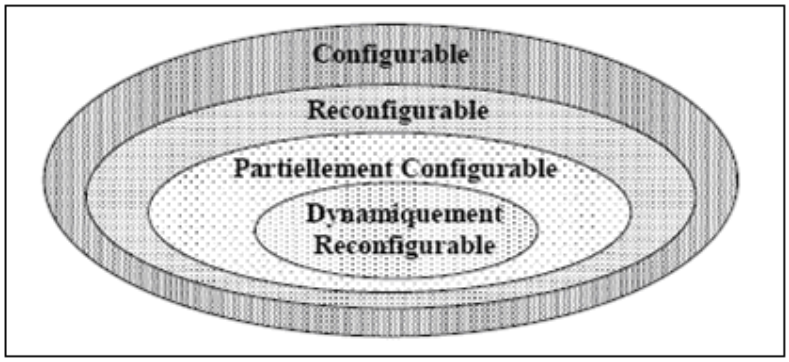

Figure 3.7 : Classification des circuits FPGAs selon leurs configurations

5.1 Circuit configurable

Un circuit Configurable est un circuit programmé et chargé par différentes données, où les interconnections d'un FPGA sont programmées afin de donner un fonctionnement spécifique pour un tel circuit.

5.2 Circuit reconfigurable

C'est le même principe de configuration citée ci-dessus, sauf que cette fois en reconfigurant le circuit FPGA une deuxième fois pour l'utiliser dans une autre fonction comme on peut garder la dernière configuration du circuit. On peut même effacer cette configuration et on configure le circuit une nouvelle fois.

5.3 Circuit partiellement reconfigurable

Un dispositif ou un circuit est défini comme partiellement reconfigurable, dans la littérature on trouve aussi la terminologie Run Time Reconfiguration RTR globale, s'il est possible de le reconfiguré sélectivement, tandis que l'état de repos du reste du dispositif est inactif, mais il conserve son information configurée. Encore, il ne semble pas y avoir n'importe quel dispositif sur le marché qui soit partiellement reconfigurable, mais non aussi dynamiquement reconfigurable, La reconfiguration partielle permet de rendre un FPGA effectif, multiple fonctions, et change des fonctions pendant le fonctionnement du système.

5.4 Circuit dynamiquement reconfigurable

Un circuit FPGA est reconfigurable dynamiquement, dans la littérature on trouve la terminologie Run Time Reconfiguration RTR locale, s'il peut être partiellement reconfiguré durant son fonctionnement, c'est-à-dire une partie du circuit correspondant à certaines fonctions logiques et leur interconnections peuvent être changées sans affecter le fonctionnement de la logique restante.

On peut aussi parler de reconfiguration dynamique dans le cas où plusieurs circuits FPGAs sont connectés entre eux et il s'agit de reconfiguré un seul composant FPGA tout en maintenant les autres circuits en fonctionnement.

6. Architecture de la famille Virtex-II

La famille Virtex -II, a une capacité de 4 à 10 millions de portes dans une technologie de pointe : 0.15µm à 8 niveaux de métallisation. Leur architecture reste très près de celle du Virtex où des blocs multiplieurs câblés font leur apparition. Cette famille contient une architecture optimisée pour une grande vitesse, horloge système interne 420 MHz, cadence externe de 33 MHz à 133 MHz et une consommation d'énergie minimale.

Elle combine entre la flexibilité dans l'intégration et la densité d'intégration. Elle comporte deux modules de base : les cores générateurs (fonctions DSP, fonctions mathématique, microprocesseur) et les modules personnalisés. Elle apporte des solutions pour les télécommunications, les réseaux, la vidéo et les applications DSP. C'est la première famille à avoir intégré des multiplieurs et des blocs Select RAM dans sa structure interne.

Son architecture est divisée en deux blocs principaux, les blocs d'entrée/sortie programmables et bloc logique interne programmable comme le montre **la figure 3.8 [DAO06].**

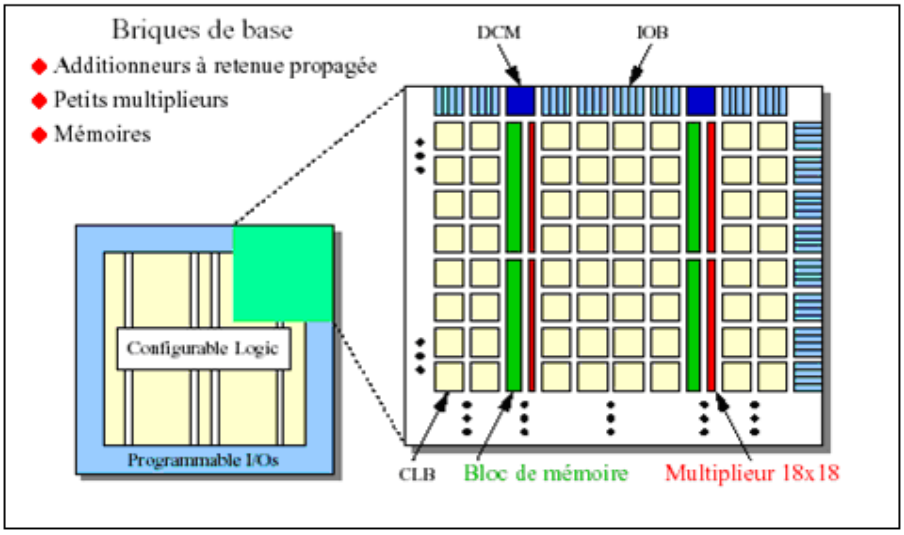

Figure 3.8 : Architecture interne de la famille VIRTEX-II

6.1 Les blocs Multiplieurs

Les blocks multiplieurs sont des multiplieurs de18x18 bits. Le circuit VIRTEX-II incorpore une grande densité de blocs multiplieurs. Chaque multiplieur peut être associé au bloc Select RAM ou peut être utilisé indépendamment **Figure 3.8**.

6.2 Les blocs DCM (Digital Clock Manager)

Le DCM produit le nouveau système d'horloges, soit intérieurement ou extérieurement au FPGA. Il produit une large gamme de fréquences de l'horloge par multiplication et même par division. Le bloc logique programmable contient plus de 12 DCM (**Figure 3.8)**.

7. Kit de développement RC203E de Celoxica [CEL10]

La famille FPGA Virtex-II possède les outils avancés pour répondre à la demande des applications de haute performance. Le kit de développement RC203E de Celoxica fournit une excellente plateforme pour explorer ces outils. L'utilisateur peut alors utiliser toutes les ressources disponibles avec rapidité et efficacité.

Un diagramme simplifié de la carte de développement Celoxica RC203E est donné par **la figure 3.9**.

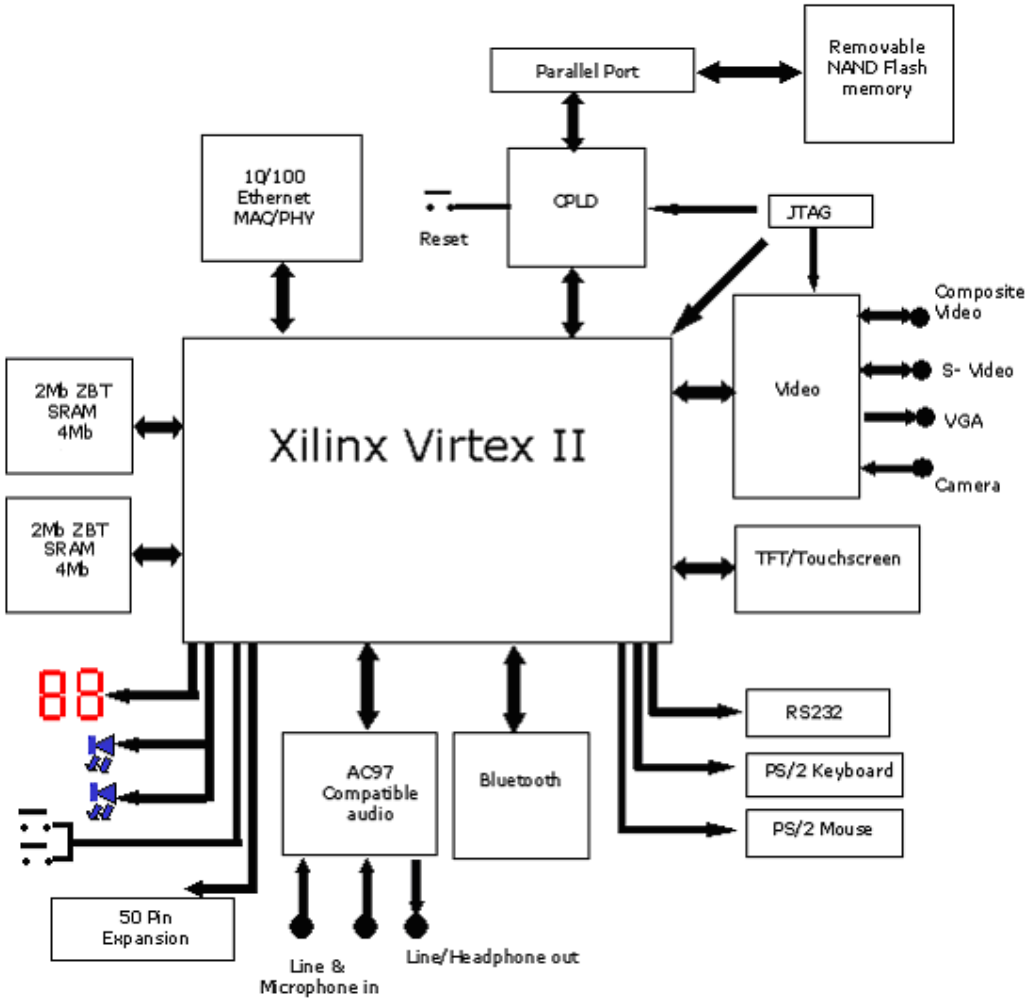

Figure 3.9 : Diagramme de la carte de développement RC203E de Celoxica

La carte de développement RC203E contient le circuit FPGA **XC2V3000-4FG456C**. Ce circuit fait partie de la famille Virtex-II, qui est une famille de circuits développés pour

des applications haute performance telles que les télécommunications, l'imagerie et les applications DSP. Il possède 50 broches dont 33 peuvent être utilisées en entrées/sorties.

La carte contient également deux mémoires SRAM de 4MB. Elle présente 2 générateurs d'horloges internes. Elle contient aussi un circuit de remise à zéro « Reset » activé par un bouton poussoir.

Deux LEDs et deux afficheurs 7 segments à cathode commune sont présents sur la carte, et peuvent être utilisés durant la phase de test et de debugging. Il existe aussi 2 entrées exploitables par l'utilisateur (DIP Switch) qui peuvent être mis statiquement à un état haut ou bas.

La carte possède une interface RS232, une interface JTAG pour programmer l'ISP PROM et configurer le circuit FPGA ainsi qu'un connecteur de câble parallèle qui peut aussi être utilisé pour configurer le FPGA.

Il existe également un port VGA, un port camera, un port de composition vidéo IN/OUT, une interface audio de type AC'97 compatible, un connecteur clavier PS/2, un connecteur sourie, et une mémoire flash (Smart Media flash Memory) pour charger le fichier bit.

La carte de développement comporte aussi un circuit programmable CPLD pour la configuration/reconfiguration de la carte, un connecteur Ethernet MAC/PHY 10/100 BaseT et trois générateurs de tension (12v, 5v, 3.3v).

8 Le langage de programmation du FPGA : le Handel-C [JUL01]

Le langage Handel-C est développé par l'université d'Oxford, en Grande-Bretagne. Il permet de générer des circuits, mais n'est pas un langage de description de circuits, c'est un langage séquentiel. Les programmes écrits en Handel-C peuvent d'ailleurs être traduits en C-ANSI à l'aide d'un compilateur disponible, et exécutés sur des machines séquentielles. Le compilateur produit une netlist qui doit être ensuite être appliquée aux circuits reconfigurables à l'aide d'outils software dédiés. Pour obtenir un circuit performant, il est nécessaire de paralléliser. Les outils de parallélisation automatique ne sont pas suffisamment performants, les concepteurs de Handel-C ont donc choisi de laisser le programmeur exprimer les différentes formes de parallélisme : parallélisme de processus, parallélisme d'assignation et parallélisme d'expression.

Handel-C a permis de réaliser des applications impressionnantes, comme des jeux vidéo sans ordinateur, en branchant un moniteur VGA directement en sortie d'un circuit Xilinx qui pilotait à la fois l'affichage et le jeu. Le même programme, compile sur une Sun Sparc-5 s'exécute 5 fois plus lentement.

C'est en fait un langage de programmation plutôt qu'un HDL. C'est un langage qui permet de générer des circuits logiques, sans être pour autant un HDL (Hardware Description Langage). Il est plutôt destiné à compiler des algorithmes de haut niveau pour en faire des portes logiques (niveau hardware), sans que l'utilisateur ne se préoccupe de savoir exactement comment fonctionne le calculateur. En un sens, le Handel-C est au hardware ce qu'un langage de haut niveau conventionnel est à l'assembleur.

8.1 Les programmes en Handel-C

La syntaxe du Handel-C est basée sur celle du C dont elle reprend de nombreuses structures. Aussi les programmes en Handel-C sont-ils intrinsèquement séquentiels. L'ordre d'écriture des instructions correspond exactement à leur ordre d'exécution. Bien sur, comme tout langage qui se respecte, le Handel-C contient des structures de contrôle du flot du programme. Par exemple, l'exécution d'une partie de code peut être conditionnée par la valeur d'une expression, ou un bloc de code peut-être répété un certain nombre de fois grâce à une boucle.

Il est important de noter que les circuits logiques générés par le Handel-C sont exactement ceux nécessaire à l'exécution du programme. Il n'y a pas d'interpréteur comme en assembleur ; les portes logiques constituant le circuit Handel-C final sont des instructions assembleur du système Handel-C.

8.2 Les programmes parallèles

Puisque le compilateur Handel-C génère du hardware de bas niveau, il est possible d'accroître les performances du système en utilisant le parallélisme. Bien que le séquentiel soit inhérent au Handel-C, on peut, et c'est même parfois primordial, demander au compilateur de créer du hardware pour exécuter des instructions en parallèle. Et il s'agit d'un vrai parallélisme : deux instructions seront exécutées exactement au même instant par deux structures du hardware bien distinctes, contrairement aux ordinateurs classiques qui en donnent seulement l'illusion **(figure 3.10)**.

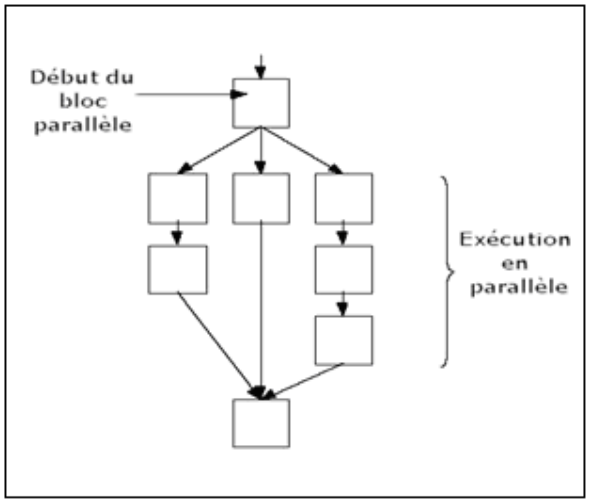

Figure 3.10 : Séparation et regroupage des branches parallèles

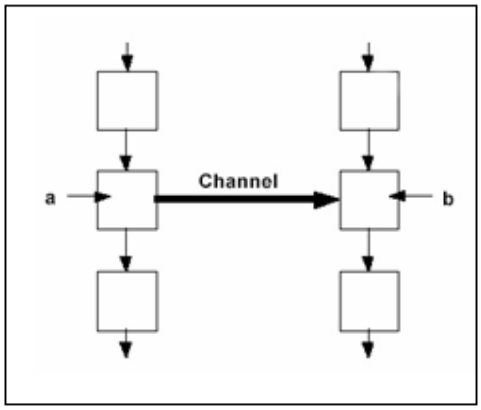

Figure 3.11 : Communication par canal entre branches parallèles.

Si une branche arrive au niveau du canal avant l'autre, elle attend jusqu' à ce que cette dernière soit elle aussi prête pour le transfert de données.

Quand un bloc parallèle est rencontré, chacune de ses branches est exécutée simultanément au départ du bloc en question. Les flots d'exécution parallèles se rejoignent à la fin du bloc quand toutes les branches ont été exécutées. Toute branche se terminant "très rapidement" est obligée d'attendre la plus lente avant de continuer,c'est-à-dire avant que l'on ne sorte du bloc parallèle.

8.3 Les canaux de communication

Les canaux constituent le lien entre les branches parallèles. Une des branches fournit des données sur le canal tandis que l'autre la lit. Les canaux servent aussi à

synchroniser les différentes branches parallèles car le transfert de données ne peut avoir lieu que si chacune des branches est prête pour le faire : si la branche émettrice n'est pas prête pour la communication alors la réceptrice doit attendre, et vice versa (**figure 3.11**).

8.4 Portée et partage des variables

La portée des déclarations est, comme en C traditionnel, basée autour des blocs de code, un bloc étant encadré par des accolades. En gros, cela signifie, d'une part, que les variables globales doivent être déclarées en dehors de tout bloc et, d'autre part, qu'une variable déclarée dans un bloc sera valide dans celui-ci ainsi que dans tous ses sous blocs. **La figure 3.12** nous en donne une illustration.

Puisque les structures parallèles sont de simples blocs de code, une variable peut-être valide dans deux branches parallèles : il suffit par exemple d'avoir un bloc principal ou on déclare sa variable et d'avoir deux sous blocs parallèles. Ceci peut entraîner des conflits de ressource si plusieurs sous blocs parallèles accèdent en même temps à la variable en question. La syntaxe du Handel-C stipule qu'une variable ne peut pas être accédée par plus d'une branche parallèle à la fois.

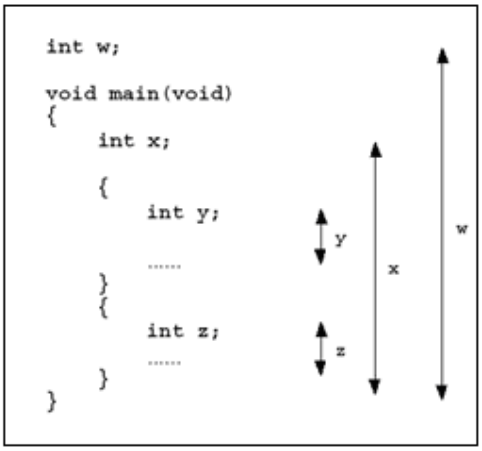

Figure 3.12 : Domaine de validité des variables

8.5 Différences fondamentales entre le Handel-C et le C

Lors du portage d'un programme du C vers le Handel-C, il convient de faire attention, car le Handel-C, bien qu'il soit basé sur la syntaxe du C, comporte tout de même quelques particularités importantes. La liste ci-dessous en pressente les principales :

✓ Les flottants (nombres à virgules) n'existent pas en Handel-C. Il n'y a que des entiers.

✓ Les pointeurs n'existent plus.

✓ Dans un cycle d'horloge, les RAM et les ROM ne peuvent avoir qu'une seule entrée accédée. Ainsi deux branches parallèles ne peuvent pas accéder à une RAM en même temps par exemple.

✓ Lors de l'accès des tableaux, l'index doit être une constante définie explicitement avant la compilation.

Ayant un tableau tab, il est donc impossible d'écrire une boucle sur une variable n pour accéder à tab[n].

Conclusion Etant donné que nous devons porter un programme du C vers le Handel-C, il nous faudra donc trouver des moyens de contourner les restrictions ci-dessus.

8.6 Comparaison Handel-C/VHDL

1. Le langage Handel-C, est intéressant pour les ingénieurs en software, qui le plus souvent ne maitrisent pas l'aspect matériel des circuits FPGA, ce qui leur permet de développer et d'implémenter des architectures sur ces circuits **[KOB09].**

2. Le Handel-C présente un certain intérêt pour les utilisateurs non initiés au matériel et aux circuits FPGA et nécessite beaucoup moins de temps pour le développement des implémentations, en revanche les architectures développées sous Handel-C ne sont pas optimisées, elles consommèrent plus de ressources et fonctionnèrent à des fréquences moins rapides **[KOB09].**

Le tableau suivant présente une comparaison entre le Handel-C et le VHDL : **[RAJ09]**

Handel-C	VHDL
C'est un langage de programmation matériel à niveau d'abstraction élevé.	C'est un langage de description matériel. Conçu exclusivement pour des ingénieurs de matériel.
Prototypage rapide et optimisation au niveau haut.	Les programmes sont fortement optimisés.
Les problèmes de bas niveau sont pris par le compilateur, le travail d'un concepteur est à un niveau d'abstraction plus élevé.	Prévoit que le concepteur doit être au courant de matériel de bas niveau.
La simulation est rapide, beaucoup de constructeurs encourage le concepteur à essayer plusieurs solutions alternatives.	Le conception est lourde pour essayer beaucoup de stratégies d'implémentation.

Tableau 3.1: Table comparatif entre Handel-C et VHDL

9. Etapes nécessaires au développement d'un projet sur FPGA

Le développement en Handel-C nécessite l'utilisation de deux outils : le simulateur et le synthétiseur. Le premier va nous permettre de simuler notre description ; cet outil interprète directement le langage Handel-C et il comprend l'ensemble du langage. L'objectif du synthétiseur est très différent, il doit traduire le comportement décrit en Handel-C en fonctions logiques de bases, celles-ci dépendent de la technologie choisie ; cette étape est nommée « synthèse ». L'intégration finale dans le circuit cible est réalisée par l'outil de placement et routage. Celui-ci est fourni par le fabricant de la technologie choisie.

Le langage Handel-C permet d'écrire des descriptions d'un niveau comportemental élevé. La question est de savoir si n'importe quelle description comportementale peut être traduite en logique ?

Avec les outils actuels, il est possible de disposer des fichiers résultat pour chaque étape. Le même fichier de simulation est ainsi utilisable pour vérifier le fonctionnement de la description à chaque étape de la procédure de développement. **La figure 3.13** donne les différentes étapes nécessaires au développement d'un projet sur circuit FPGA. **[MEC06]**

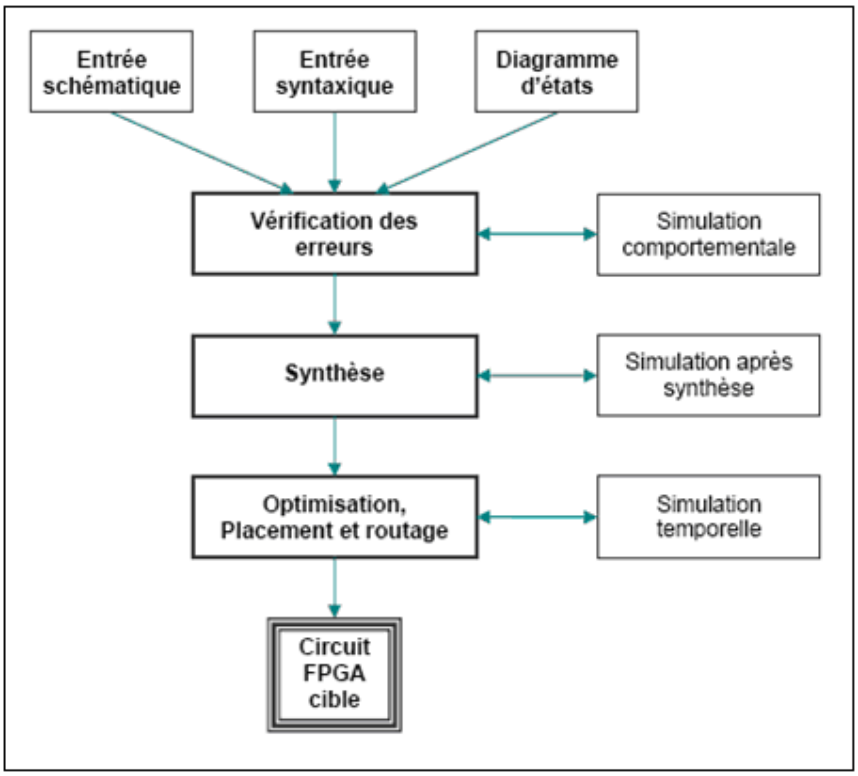

Figure 3.13 : Organisation fonctionnelle de développement d'un projet sur circuit FPGA

9.1 Saisie du texte Handel-C

La saisie du texte Handel-C se fait dans le logiciel «DK». Ce logiciel propose une palette d'outils permettant d'effectuer toutes les étapes nécessaires au développement d'un projet sur circuit FPGA. Il possède également des outils permettant de mettre au point une entrée schématique ou de créer des diagrammes d'état, qui peuvent être utilisés comme entrée au lieu du texte Handel-C.

La figure 3.14 montre l'interface du logiciel «DK».

La saisie du texte se fait sur la partie droite de l'écran, on voit en haut à gauche la hiérarchie du projet.

Il faut commencer par créer un projet, ensuite inclure des fichiers sources dans lesquels il faut saisir le texte Handel-C désiré. On peut inclure autant de sources qu'on veut dans un projet.

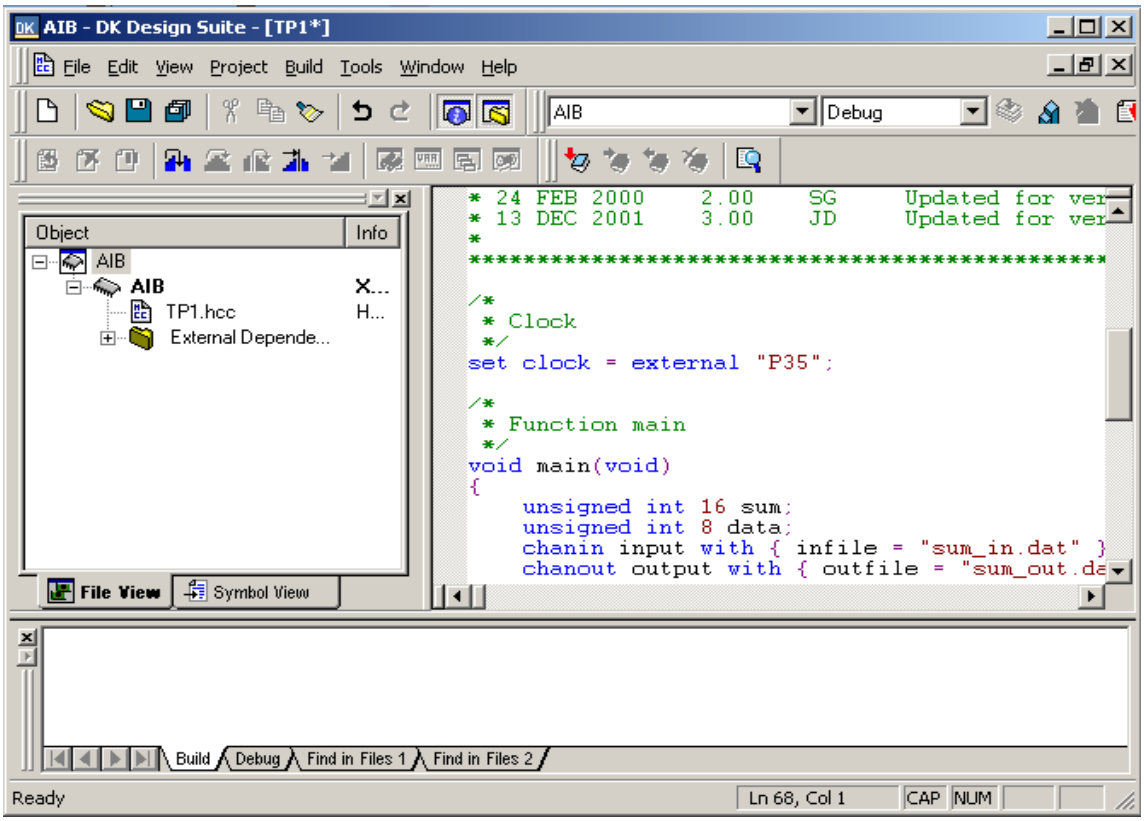

Figure 3.14 : Vue d'ensemble du logiciel « DK »

9.2 Vérification des erreurs

Cette étape est effectuée en appuyant sur le bouton « CTRL+F7 » ou sur le menu Build→compile. Elle permet de vérifier les erreurs de syntaxe du texte Handel-C et d'afficher les différentes alarmes « warnings » liées au programme. S'il y'a des erreurs dans le programme, il ne peut pas être synthétisé, mais la présence d'alarmes n'empêche pas de poursuivre normalement les autres étapes du développement.

Contenant la description, cette étape permet donc de valider la syntaxe du programme et de générer la « netlist », qui est un fichier de l'application sous forme d'équations logiques.

9.3 Synthèse, Optimisation, placement et routage

La synthèse permet de réaliser l'implémentation physique d'un projet. Le synthétiseur a pour rôle de convertir le projet, en fonction du type du circuit FPGA cible utilisé, en portes logiques et bascules de base. Dans notre cas le résultat de l'étape de synthèse est un fichier (Bit). Donc on va choisir sur DK le type de fichier de sortie puis on démarre la synthèse à partir de Build→Build nom de projet **(figure 3.15).**

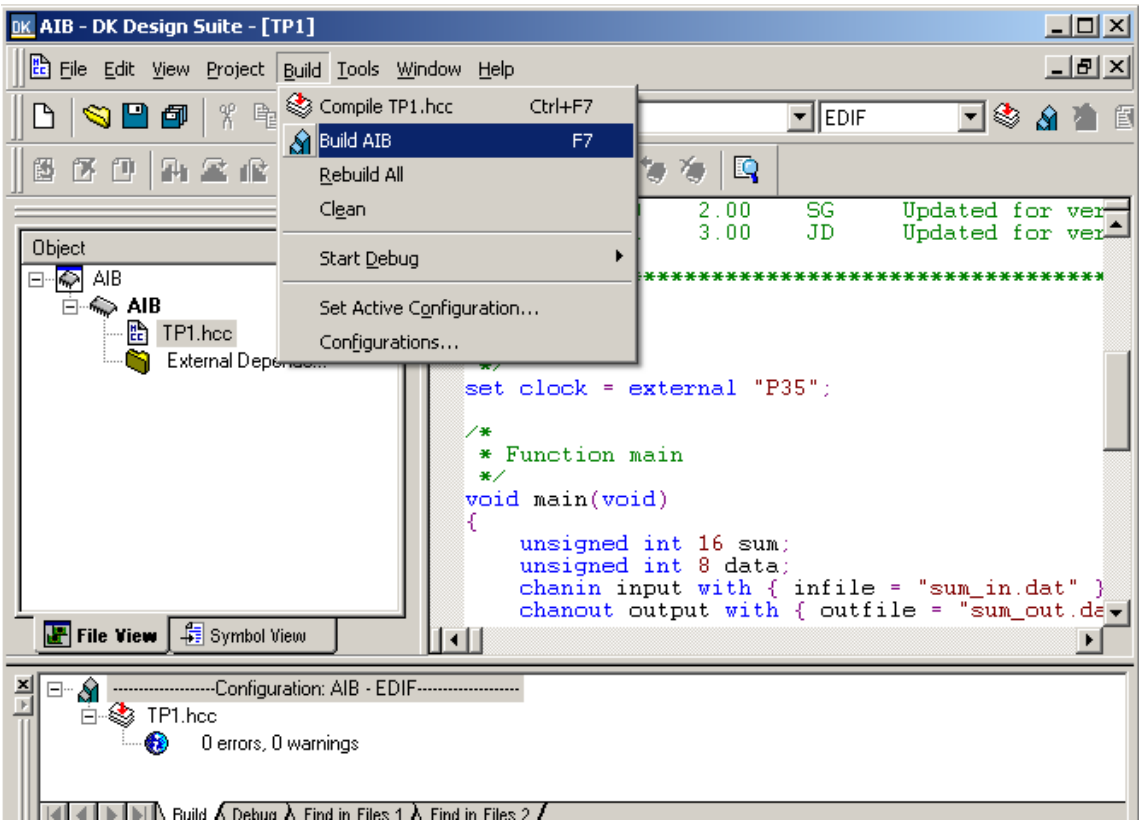

Figure 3.15 : Aperçu de l'étape de synthèse sur DK

9.4 Simulation

L'outil de simulation utilisé est le « PAL Virtual Platform » (**figure 3.16**). La simulation permet de vérifier le comportement d'un design avant implémentation dans le composant cible.

Elle représente une étape essentielle qui nous fera gagner du temps lors de la mise au point sur la carte. Il faut juste noter qu'un projet peut être simulé même s'il n'est pas synthétisable.

Lors de l'étape de simulation comportementale, on valide l'application indépendamment de l'architecture et des temps de propagation du futur circuit cible. La phase de simulation après synthèse valide l'application sur l'architecture du circuit cible, et enfin la simulation temporelle prend en compte les temps de propagation des signaux à l'intérieur du circuit FPGA cible.

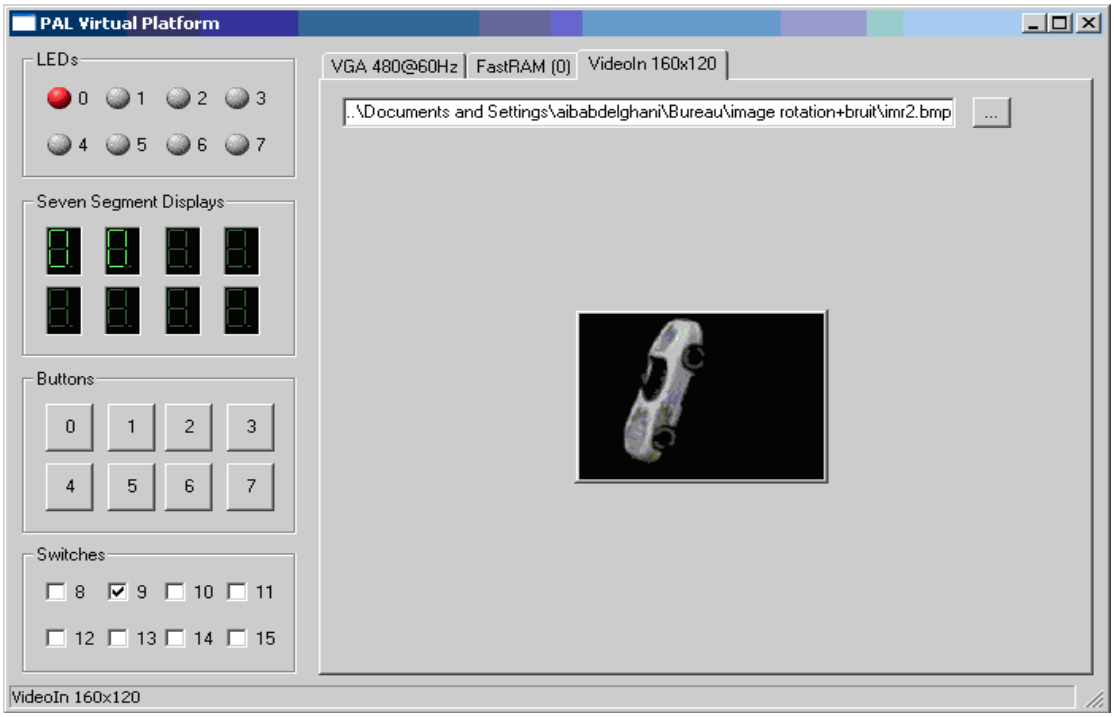

Figure 3.16 : L'outil de simulation « PAL Virtual Platform »

9.5 Programmation du composant et test

Dans cette dernière étape, on utilise FTU2 (File Transfert Utility) pour charge le fichier (.bit) sur le circuit FPGA à travers le port parallèle. Une fois le programme chargé sur le circuit, on peut tester et visualiser les résultats directement sur la carte de développement RC203E à travers les nombreuses interfaces qu'elle offre.

La figure suivante présente l'outil de transfert FTU2 :

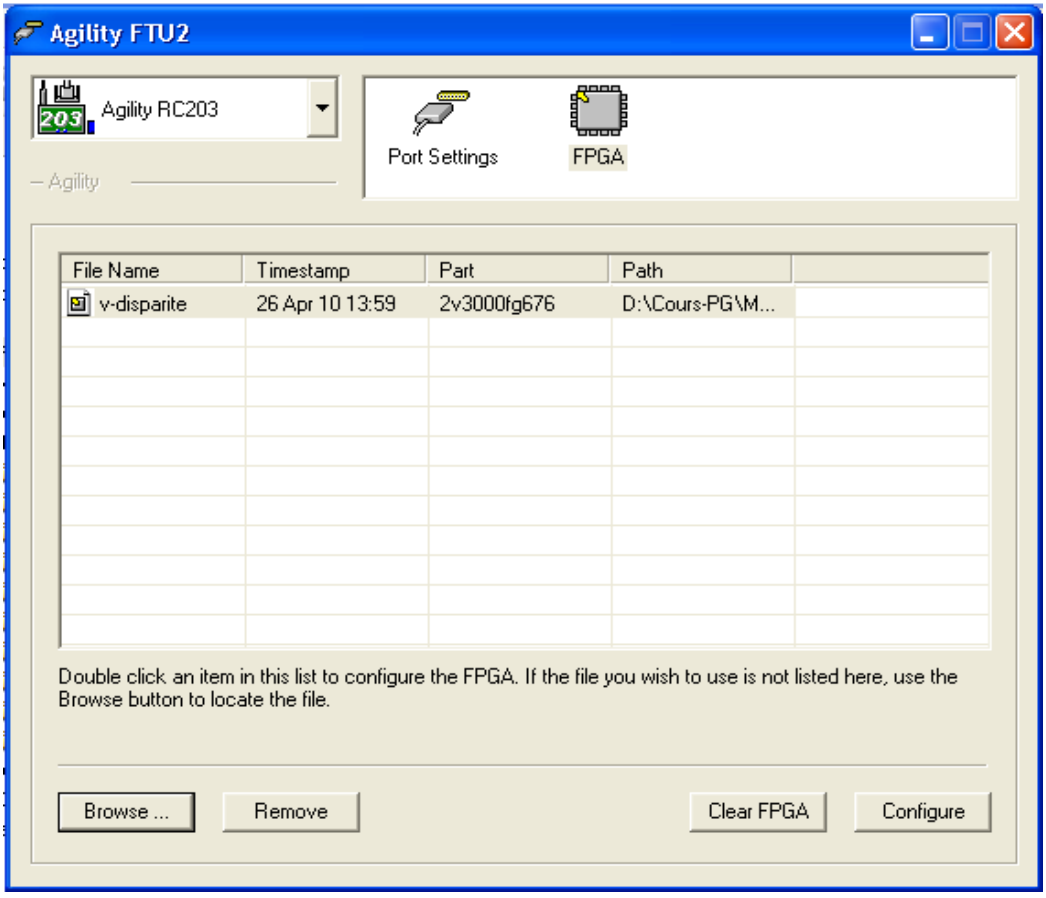

Figure 3.17: LE FTU2 (File Transfert Utility)

10. Conclusion

Dans ce chapitre, nous présentons différents circuits numériques utilisés pour la mise en œuvre des algorithmes de traitement d'image en temps réel. Par la suite, nous présentons un aperçu sur les circuits FPGAs en indiquant que leurs caractéristiques de reconfiguration et leurs souplesses fournissent un avantage indéniable sur les autres composants. En dernier lieu, nous terminons ce chapitre par une présentation générale de la plateforme de développement matériel et logiciel. Une comparaison entre les différentes approches de développement à été faite, ce qui ma permis d'opter pour le langage Handel-C qui ce révèle adapté à l'implémentation des algorithmes de traitement des images.

Dans le chapitre suivant, nous présenterons l'asservissement visuel des robots mobiles et nous nous intéresserons, particulièrement, aux différents types d'asservissement visuel, les différents modèles des robots mobiles, leurs modèles cinématiques et la synthèse d'une commande stabilisant

Asservissement visuel des robots mobiles

1. Introduction

L'un des plus grands rêves de l'Homme est de pouvoir faire exécuter par des acteurs autres que lui-même des tâches ou activités qu'il considère comme aliénantes, dangereuses ou simplement ennuyeuses et non gratifiantes. L'apparition progressive de machines puis de la notion d'automatisation a permis de modifier la nature des travaux effectués par les hommes, donc de déporter cette aspiration vers des réponses issues des progrès des techniques et des sciences. Ainsi, l'émergence de la robotique au cours des années 1960 a su apporter de nouvelles réponses à cette attente. Avec les progrès constants dans ce domaine, les robots tendent à accomplir des tâches de plus en plus complexes avec une intervention moindre de l'homme pour les guider. Ils deviennent ainsi plus autonomes, et ils interagissent de mieux en mieux avec leur environnement pour accomplir la mission qui leur a été assignée, le robot devient alors une "machine intelligente".

Vers la fin des années 1970, la robotique devient un thème de recherche à part entière stimulé par la robotique manufacturière et les travaux sur les bras manipulateurs. Mais, c'est surtout à partir des années 1980 qu'un intérêt croissant a été accordé à la recherche en robotique mobile motivé d'une part par l'augmentation de la puissance de calcul embarquée et l'amélioration des capteurs (miniaturisation, robustesse, etc.) ; et d'autre part par la plus grande diversité d'applications offertes. Toutefois, il apparaît rapidement que, si pour la plupart des bras manipulateurs, dotés en général d'un nombre important de degrés de liberté, à toute configuration de l'espace des variables articliers correspond une situation exécutable dans l'espace réel, cette propriété fondamentale n'est souvent pas valable pour quelques types de robots mobiles. En effet, certains robots mobiles, à roues, ne peuvent pas se déplacer dans l'axe de leurs roues du fait de la contrainte non-holonome de roulement sans glissement, problème bien connu en mécanique. On souhaite alors faire effectuer aux robots mobiles des mouvements permettant de tenir compte de ces contraintes mécaniques.

La problématique de la navigation et du pilotage des robots, longtemps cantonnée au domaine de la robotique de manipulation, s'ouvre aujourd'hui vers de nombreux autres champs d'application, tels que : les véhicules, les drônes aériens, les engins sous-marins, la chirurgie, etc. Dans tous les cas, il s'agit de faire évoluer des systèmes de façon sûre dans des milieux imparfaitement connus en contrôlant les interactions entre le robot et son environnement. Ces interactions peuvent prendre différents aspects : actions de la part du système robotique (se positionner par rapport à un objet, manœuvrer pour se garer, etc.), réactions vis-à-vis d'événements provenant du monde qui l'entoure (éviter des obstacles, poursuivre une cible mobile, etc.). Le degré d'autonomie du robot réside alors dans sa capacité à prendre en compte ces interactions à tous les niveaux de la tâche robotique. En premier lieu, lors de la planification, cela se fera par l'acquisition, la modélisation et la manipulation des connaissances sur l'environnement et sur l'objectif à réaliser. Ensuite, durant

l'exécution, il s'agira d'exploiter les données perceptuelles pour adapter au mieux le comportement du système aux conditions de la mission qu'il doit réaliser. Ainsi, classiquement, une tâche de navigation se décompose en plusieurs phases :

- **Perception** : selon l'objectif fixé et les difficultés rencontrées, le système de navigation acquiert les informations nécessaires pour l'accomplissement de la tâche ;
- **Modélisation** : le robot procède à l'analyse de ces données perceptuelles afin de produire une représentation interprétable de l'environnement ;
- **Localisation** : pour décider des déplacements, le robot doit déterminer sa situation par rapport au modèle produit ;
- **Planification** : le robot décide de l'itinéraire, puis détermine la trajectoire ou les mouvements nécessaires, ainsi que les conditions de leur réalisation ;
- **Action** : le robot doit s'asservir sur les déplacements décidés pour l'accomplissement de sa mission.

La perception et la modélisation de l'environnement en vue de la planification des actions est un thème majeur en robotique **[LAU 01]**. Il se décline en de nombreuses variantes selon la connaissance a priori dont on dispose : elle peut être complète permettant ainsi la planification hors-ligne de la tâche, ou bien absente, nécessitant alors une acquisition en ligne du modèle durant une phase d'exploration.

De même qu'il est important de prendre en compte le processus de perception très tôt dans la description de la tâche, il est tout aussi indispensable de contrôler l'interaction entre le robot et son environnement lors de son exécution. Cela se traduit par la prise en compte explicite d'informations perceptuelles, d'une part, dans la constitution de boucles de commande robustes et, d'autre part, dans la détection d'événements externes nécessitant une modification du comportement du robot. Dans les deux cas, il s'agit de rendre robuste le système face à une certaine variabilité des conditions d'exécution de la mission. Cette variabilité peut provenir d'erreurs de mesures ou de modèle, dues aux capteurs ou au système commandé. Mais elle peut également provenir du monde dans lequel le système évolue qui peut être incertain ou dynamique. Pour pallier à ces incertitudes, certains auteurs **[KHA 97]** proposent des solutions permettant de déformer la trajectoire initialement planifiée, de manière à éviter les obstacles non prévus. Cependant, ce genre d'approche, qualifiée de globale, se base sur la modification d'un chemin ou d'une trajectoire préalablement établis. Par conséquent, elles requièrent la construction d'une carte pour générer l'itinéraire, ainsi que la localisation du robot par rapport à celui-ci. D'autres types d'approches, dite locales, que nous considérons dans notre étude, permettent d'intégrer directement les informations sensorielles dans la boucle de commande. Sur la base de ces informations l'objectif est alors de synthétiser un asservissement qui guide le robot vers son but tout en tenant compte des variations du milieu où évolue le système, sans connaissance a priori sur celui-ci. Toutefois, ne disposant pas d'un modèle de l'environnement, les méthodes locales souffrent généralement des problèmes de

minima locaux : c'est-à-dire des configurations où la commande appliquée est nulle, bien que le robot ne se trouve pas encore à la situation désirée. Néanmoins cette approche permet la synthèse de lois de commande très réactives, précises et robustes.

2. Commande référencée vision

La commande référencée vision **[COR 96]** consiste à contrôler les mouvements d'un système robotique en utilisant des informations visuelles, notées s, issues d'une ou plusieurs caméras (ou plus généralement d'un capteur de vision) embarquées ou non sur le système. De nombreux travaux sont basés sur l'exploitation de ces données pour réaliser différents objectifs tels que le positionnement face à un objet, son suivi, sa saisie, etc. La première utilisation de la vision en boucle fermée est revient à SHIRAI et INOUE [**SHI 73]** qui décrivirent comment un capteur de vision pouvait augmenter la précision du positionnement. On parlait alors de retour par vision (Visual feedback). Mais c'est à HILL et PARK [**HIL 79]** que l'on doit l'apparition du terme asservissement visuel (Visual servoing). Plusieurs approches ont depuis vu le jour. [**SAN 80]** en propose ainsi deux grandes classes selon l'utilisation des informations visuelles : l'asservissement visuel 3D (ou position-based control) et l'asservissement visuel 2D (ou image-based control). Notons qu'il existe aussi des approches intermédiaires plus récentes telles que l'asservissement visuel 2D1/2 **[MAL 98]** ou d2D/dt **[CRÉ 98]**. Nous présentons ci-après les structures de commande 3D et 2D1/2, avant de détailler l'asservissement visuel 2D que nous avons plus particulièrement considéré dans le cadre de nos travaux.

2.1 Asservissement visuel 3D

L'asservissement visuel 3D utilise en entrée de la boucle de commande des informations tridimensionnelles, à savoir la situation r de la caméra, par rapport à l'objet d'intérêt. La tâche à réaliser s'exprime alors sous la forme d'une situation de référence r^* à atteindre (**figure 4.1**). La commande repose ainsi sur la détermination de la situation r de la caméra, à partir des informations visuelles extraites de l'image **[Dav 07] (figure 4.2).**

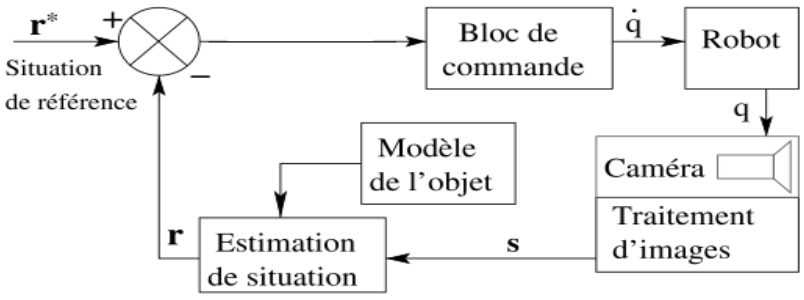

Figure 4.1 Schéma bloc Asservissement visuel 3D

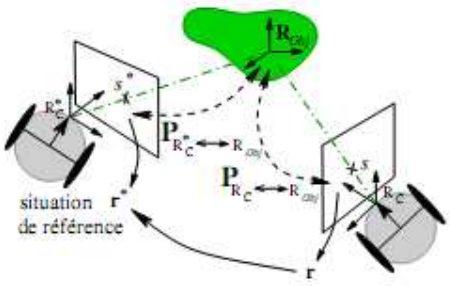

Figure 4.2 Schéma de principe Asservissement visuel 3D

De nombreuses méthodes permettent d'estimer la situation d'une caméra par rapport à un objet à partir de l'image perçue de cet objet. Elles reposent très généralement sur la connaissance a priori d'un modèle (3D ou 2D) de l'objet et des paramètres intrinsèques de la caméra. Ces méthodes utilisent des informations visuelles de différentes natures, telles que des points **[HOR 89]**, des droites **[DHO 89]** Toutefois des travaux, issus des recherches en reconstruction 3D par vision dynamique, permettent d'estimer le modèle de l'objet d'intérêt, ou de localiser la caméra à partir de mesures de mouvement 2D ou 3D, élargissant ainsi les tâches considérées.

Finalement, l'asservissement visuel 3D s'exprimant directement dans l'espace des configurations permet la définition de lois de commande extrêmement simples pour aller itérativement d'une situation à une autre. Toutefois, l'inconvénient qui en résulte est qu'aucun contrôle véritable dans l'image n'est effectué ce qui implique que l'objet peut très bien être perdu. De plus, cette approche requiert une interprétation du motif visuel pour caractériser la situation du robot, et nécessite donc une reconstruction de l'état du système. On y retrouve alors les mêmes problèmes d'incertitude et de précision des schémas de commande des robots mobiles classiques.

2.2 Asservissement visuel 2D1/2

L'asservissement visuel 2D1/2 **[MAL 98]** exploite des informations à la fois de nature 2D et 3D. Cette technique est basée sur l'estimation de l'homographie, notée H, qui relie l'image d'au moins trois points entre différents plans projectifs **[FAU 93]** (voir **figure 4.3)**. L'homographie est une application projective bijective, correspondant à une transformation linéaire entre deux plans projectifs. Plus précisément, elle permet d'établir (à un facteur d'échelle α près) une bijection entre un objet de l'espace 3D et une image 2D, ou bien entre deux images 2D d'un même objet. Dans ce dernier cas, elle permet de lier les projections d'un ensemble de points 3D P_i, appartenant à un même plan, sur deux plans image Π_1 et Π_2 par la relation :

$$p_i/\Pi_2 = \alpha H p_i/\Pi_1, \forall \alpha \in R^* \qquad (4.1)$$

où P_i/Π_1 et P_i/Π_2 représentent respectivement la projection des points P_i sur les plans Π_1 et Π_2. Dans le cas d'un objet plan, un minimum de quatre points appariés sur les images courantes et désirées permet l'estimation de l'homographie par la résolution d'un système linéaire. Dans le cas d'un objet quelconque, un appariement de huit points est nécessaire.

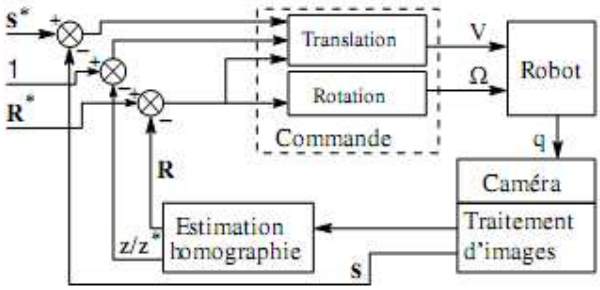

Figure 4.3 Schéma bloc Asservissement visuel 2D1/2

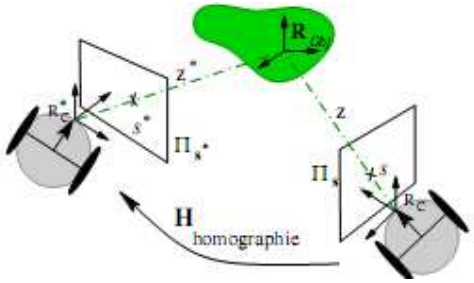

Figure 4.4 Schéma de principe Asservissement visuel 2D1/2

Ainsi, l'asservissement visuel 2D1/2 repose sur la détermination de l'homographie H entre l'image courante et l'image désirée **[Dav 07]**. En effet, à partir de cette homographie il est possible de calculer le déplacement en rotation **R** que la caméra doit effectuer pour atteindre la situation spécifiée, ainsi que la direction de son déplacement en translation. Cette matrice **H** fournit également le rapport **z/z*** entre les distances courante et désirée de la caméra à l'objet (**figure 4.3**). Comme la translation à réaliser n'est connue qu'à un facteur d'échelle **α** près, un asservissement visuel purement 3D est impossible. Cependant, il est possible d'aboutir à une solution grâce à la combinaison des informations 2D (fournies par l'image) et 3D disponibles. Le vecteur des informations visuelles **s** est donc défini sur la base de donnée 2D et 3D. Ainsi, l'asservissement visuel 2D1/2 est une approche intermédiaire entre l'asservissement 3D et 2D. La boucle d'asservissement ainsi synthétisée permet de séparer la rotation et la translation de la caméra, et d'obtenir un fort découplage de la loi de commande (**figure 4.4**). L'avantage majeur de cette approche est que la connaissance du modèle géométrique 3D de l'objet n'est plus nécessaire. La seule information 3D utilisée dans la synthèse de la commande est la profondeur désirée **z*** d'un point de l'objet.

2.3 Asservissement visuel 2D

Les techniques d'asservissement visuel 2D utilisent directement les informations visuelles, notée s, extraites de l'image **[Dav 07]** (**figure 4.5**). La tâche à réaliser est alors spécifiée directement dans l'image en termes d'indices visuels de référence **s*** à atteindre. La loi de commande consiste alors à contrôler le mouvement de la caméra de manière à annuler l'erreur entre les informations visuelles courantes **s(t)** et le motif désiré **s*** (**figure 4.6**). Cette approche permet donc de s'affranchir de l'étape de reconstruction 3D de la cible et des problèmes qui y sont liés.

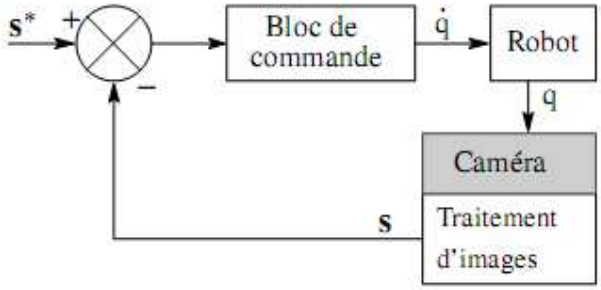

Figure 4.5 Schéma bloc Asservissement visuel 2D.

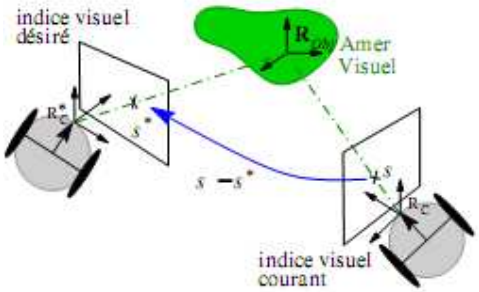

Figure 4.6 Schéma de principe Asservissement visuel 2D.

Le choix des informations visuelles et l'obtention de la relation les liant au mouvement de la caméra sont deux aspects fondamentaux de l'asservissement visuel 2D. Cette relation, obtenue par dérivation de l'information sensorielle s par rapport à la situation r de la caméra, est définie par une matrice appelée jacobienne de l'image ou matrice d'interaction.

La synthèse de la commande repose sur l'élaboration d'une méthode de calcul explicite de cette matrice souvent associée à des primitives géométriques simples, telles que des points **[FED 89],** des droites, des cercles, des ellipses **[CHA 90],** ou encore des moments de l'image **[CHA 90].** Dans ce contexte, les lois de commande synthétisées dépendent de la nature des informations visuelles choisies. Par conséquent la tâche robotique à réaliser est elle aussi dépendante de l'objet observé par la présence ou non d'informations visuelles dont on est capable de calculer la matrice d'interaction associée. C'est pourquoi, certains travaux ont proposé des

extensions, en traitant des informations visuelles pouvant caractériser l'aspect global de l'image. Ainsi, par exemple, l'exploitation des informations visuelles dynamiques **[CRÉ 98]** a conduit à l'élaboration de l'asservissement visuel d2D/dt. Néanmoins, ces asservissements visuels 2D sont, d'une manière générale, des lois de commandes relativement rapides à calculer, puisqu'il n'y a pas de phase de reconstruction 3D. Ce gain de temps n'est pas négligeable puisque le délai d'application de la commande est réduit, ce qui contribue à la stabilité du système **[WUN 97].** De plus, les asservissements basés sur l'image permettent la réalisation de tâches de manière très efficace et précise. C'est ainsi que ce type de commande se rencontre de plus en plus dans différents domaines d'application. Citons par exemple la conduite d'engins sous-marins **[RIV 97]**, de véhicules autonomes personnels **[MAR 00]** ou agricoles **[KHA 98]**. La robotique chirurgicale apparaît également comme un champ d'application privilégié des dernières avancées de l'asservissement visuel 2D **[VIT 06].**

3. Modélisation des robots mobiles

3.1 Définitions

On note $R = (O, \vec{x}, \vec{y}, \vec{z})$ un repère de référence fixe quelconque, dont l'axe $\vec{z}$ est vertical et $\grave{R} = (\acute{O}, \vec{x}, \vec{y}, \vec{z})$ un repère mobile lié au robot. On choisit généralement pour $\acute{O}$ un point remarquable de la plate-forme, typiquement le centre de l'axe des roues motrices s'il existe, comme illustré à la **figure 4.7**.

Par analogie avec la manipulation, on appelle situation **[FOU 99]** ou souvent posture **[CAM 96]** du robot le vecteur :

$$\varepsilon = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix},$$

Où x et y sont respectivement l'abscisse et l'ordonnée du point $\acute{O}$ dans R et θ l'angle $(\vec{x}, \vec{\acute{x}})$. La situation du robot est donc définie sur un espace M de dimension m = 3, comparable à l'espace opérationnel d'un manipulateur plan.

La configuration d'un système mécanique est connue quand la position de tous ses points dans un repère donné est connue **[NEI 72]**. Alors que pour un bras manipulateur cette notion est définie sans ambigüité par les positions angulaires des différentes articulations, on peut, dans le cas d'un robot mobile, donner une vision plus ou moins fine de la configuration, comme on le verra par la suite. Dans tous les cas, on définira la configuration du robot mobile par un vecteur :

$$q = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{pmatrix},$$

De n coordonnées appelées coordonnées généralisées. La configuration est ainsi définie sur un espace N de dimension n, appelée l'espace des configurations.

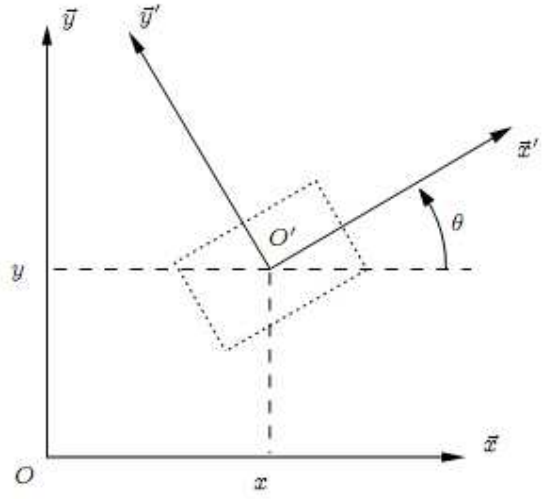

Figure 4.7 Repérage d'un robot mobile

3.2 Roulement sans glissement et contraintes non holonomes

3.2.1 Roulement sans glissement

La locomotion à l'aide de roues exploite la friction au contact entre roue et sol. Pour cela, la nature du contact (régularité, matériaux en contact) a une forte influence sur les propriétés du mouvement relatif de la roue par rapport au sol. Dans de bonnes conditions, il y a roulement sans glissement (r.s.g.) de la roue sur le sol, c'est à dire que la vitesse relative de la roue par rapport au sol au point de contact est nulle. Théoriquement, pour vérifier cette condition, il faut réunir les hypothèses suivantes :

- le contact entre la roue et le sol est ponctuel ;
- les roues sont indéformables, de rayon r.

En pratique le contact se fait sur une surface, ce qui engendre bien évidemment de légers glissements. De même, alors qu'il est raisonnable de dire que des roues pleines sont indéformables, cette hypothèse est largement fausse avec des roues équipées de pneus. Malgré cela, on supposera toujours qu'il y a r.s.g. et, par ailleurs, que le sol est parfaitement plan.

Mathématiquement, on peut traduire la condition de r.s.g. sur une roue. Soit P le centre de la roue, Q le point de contact de la roue avec le sol, ϕ l'angle de rotation propre de la roue et θ l'angle $(\vec{x}, \vec{\dot{x}})$, comme indiqué à **la figure 4.8**. La nullité de la vitesse relative $\overrightarrow{v_Q}$ roue/sol au point de contact permet d'obtenir une relation

vectorielle entre la vitesse $\overrightarrow{v_P}$ du centre P de la roue et le vecteur vitesse de rotation $\overrightarrow{\omega}$ de la roue **[BER 09]** :

$$\overrightarrow{v_Q} = \overrightarrow{v_P} + \overrightarrow{\omega} \bigwedge \overrightarrow{PQ} = \vec{0}$$

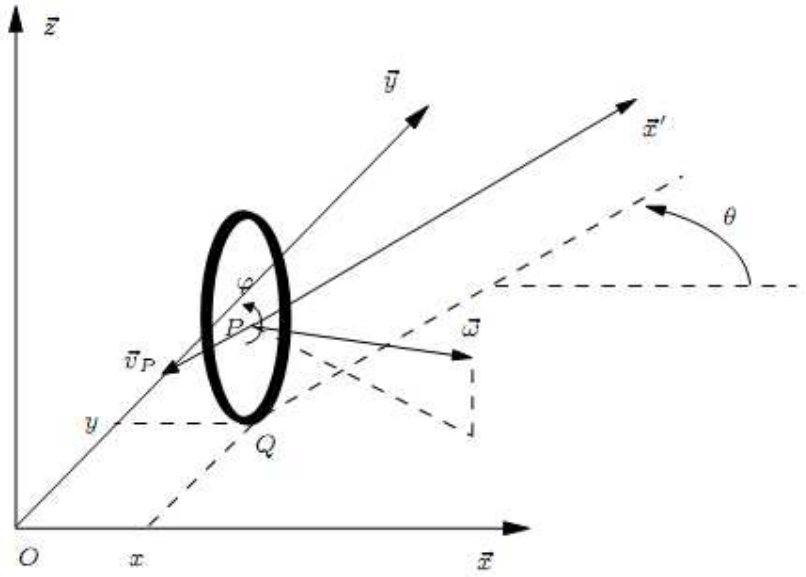

Figure. 4.8 Caractérisation du roulement sans glissement

Les points P et Q ont pour coordonnées respectives $(x\ y\ r)^T$ et $(x\ y\ 0)^T$. Il vient alors :

$$\dot{x}\vec{x} + \dot{y}\vec{y} + (\dot{\theta}\,\vec{z} + \dot{\varphi}(\sin\theta\,\vec{x} - \cos\theta\,\vec{y})) \wedge (-r\vec{z}) = \vec{0}$$

$$(\dot{x} + r\dot{\varphi}\cos\theta)\vec{x} + (\dot{y} + r\dot{\varphi}\sin\theta)\vec{y} = \vec{0}$$

Ceci nous donne le système de contraintes scalaires :

$$\dot{x} + r\dot{\varphi}\cos\theta = 0 \qquad (\mathbf{4.2})$$

$$\dot{y} + r\dot{\varphi}\sin\theta = 0 \qquad (\mathbf{4.3})$$

Que l'on peut transformer pour faire apparaitre les composantes de vitesse dans le plan de la roue d'une part et perpendiculairement à la roue d'autre part :

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \quad (\mathbf{4.4})$$

$$\dot{x}\cos\theta + \dot{y}\sin\theta = -r\dot{\varphi} \quad (\mathbf{4.5})$$

Ces contraintes traduisent le fait que le vecteur $\overrightarrow{v_P}$ soit dans le plan de la roue et ait pour module $r\dot{\varphi}$.

3.2.2 Contraintes non holonomes

Les équations précédentes, caractérisant le r.s.g. d'une roue sur le sol, sont des contraintes non holonomes. Nous nous proposons dans ce paragraphe de préciser ce que recouvre ce terme et de caractériser les systèmes non holonomes.

Soit un système de configuration q soumis à des contraintes indépendantes sur les vitesses, regroupées sous la forme $A^T(q)\dot{q} = 0$. S'il n'est pas possible d'intégrer l'une de ces contraintes, elle est dite non intégrable ou non holonome. De manière concrète l'existence de contraintes non holonomes implique que le système ne peut pas effectuer certains mouvements instantanément. Par exemple, dans le cas de la roue, il ne peut y avoir de translation instantanée parallèlement à l'axe de la roue. Un tel déplacement nécessitera des manœuvres. De même, comme on le sait bien, une voiture ne peut se garer facilement sans effectuer de créneaux.

Il n'est pas évident de dire a priori si une contrainte est intégrable ou non. Pour cela, on a recours à l'application du théorème de Frobenius, dont une version complète pourra être trouvée dans un ouvrage de référence de géométrie différentielle **[WAR 83]** ou de commande non-linéaire **[NIJ 90].** Seule la connaissance du crochet de Lie est nécessaire à notre étude. Pour deux vecteurs bi(q) et bj(q), cet opérateur est défini par :

$$\left[b_i(q), b_j(q)\right] = \frac{\partial b_j}{\partial q} b_i - \frac{\partial b_i}{\partial q} b_j$$

3.3 Les grandes classes de robots mobiles et leurs modèles

3.3.1 Disposition des roues et centre instantané de rotation

C'est la combinaison du choix des roues et de leur disposition qui confère à un robot son mode de locomotion propre. Sur les robots mobiles, on rencontre principalement trois types de roues (voir **figure 4.9) [BER 09]** :

- les roues fixes dont l'axe de rotation, de direction constante, passe par le centre de la roue ;
- les roues centrées orientables, dont l'axe d'orientation passe par le centre de la roue ;
- les roues décentrées orientables, souvent appelées roues folles, pour lesquelles l'axe d'orientation ne passe pas par le centre de la roue.

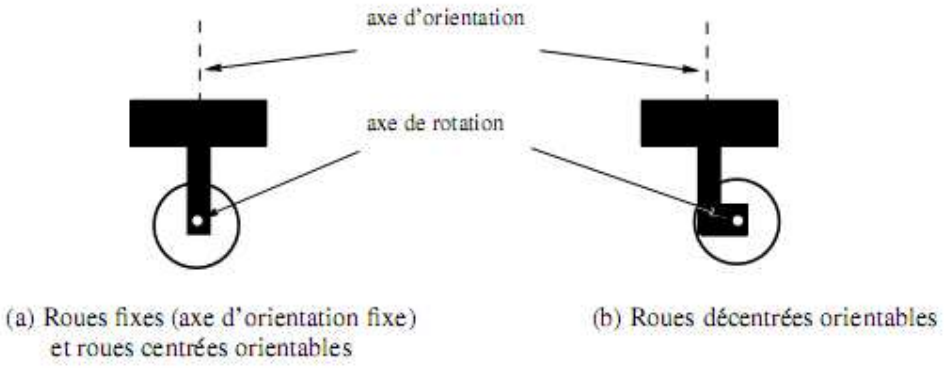

Figure 4.9 Les principaux types de roues des robots mobiles

De manière anecdotique on rencontrera aussi des systèmes particuliers, tels que les roues suédoises, les roues à plusieurs directions de roulement, etc.

Bien évidemment, pour un ensemble de roues donné, toute disposition ne conduit pas à une solution viable. Un mauvais choix peut limiter la mobilité du robot ou occasionner d'éventuels blocages. Par exemple, un robot équipé de deux roues fixes non parallèles ne pourrait pas aller en ligne droite ! Pour qu'une disposition de roues soit viable et n'entraine pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot de façon instantanée. Ce point, lorsqu'il existe, est appelé centre instantané de rotation (CIR). Les points de vitesse nulle liés aux roues se trouvant sur leur axe de rotation, il est donc nécessaire que le point d'intersection des axes de rotation des différentes roues soit unique. Pour cette raison, il existe en pratique trois principales catégories de robots mobiles à roues, que l'on va présenter ci-dessous.

3.3.2 Robots mobiles de type unicycle

a) Description

On désigne par unicycle un robot actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Le schéma des robots de type unicycle est donné à **la figure 4.10**. On y a omis les roues folles, qui n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées.

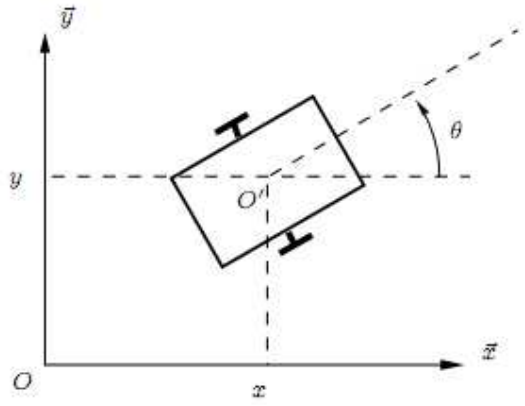

Figure 4.10 Robot mobile de type unicycle

Ce type de robot est très répandu en raison de sa simplicité de construction et de propriétés cinématiques intéressantes.

b) Modélisation

Centre instantané de rotation Les roues motrices ayant même axe de rotation, le CIR du robot est un point de cet axe. Soit ρ le rayon de courbure de la trajectoire du robot, c'est-à-dire la distance du CIR au point $\acute{O}$ (**figure 4.11**). Soit L l'entre-axe et ω la vitesse de rotation du robot autour du CIR. Alors les vitesses des roues droite et gauche, respectivement notées $v_d$et v_g et définies à la **figure 4.11**, vérifient :

$$v_d = -r\dot{\varphi}_d = (\rho + L)\omega \qquad (\mathbf{4.6})$$

$$v_g = r\dot{\varphi}_g = (\rho - L)\omega \qquad (\mathbf{4.7})$$

Ce qui permet de déterminer ρ et ω à partir des vitesses des roues :

$$\rho = L\frac{\dot{\varphi}_d - \dot{\varphi}_g}{\dot{\varphi}_d + \dot{\varphi}_g} \qquad (\mathbf{4.8})$$

$$\omega = -\frac{r(\dot{\varphi}_d + \dot{\varphi}_g)}{2L} \qquad (\mathbf{4.9})$$

L'équation (4.8) permet de situer le CIR sur l'axe des roues. Par ailleurs ces équations expliquent deux propriétés particulières du mouvement des robots de type unicycle : si $\dot{\varphi}_d = -\dot{\varphi}_g$ le robot se déplace en ligne droite ; si $\dot{\varphi}_d = \dot{\varphi}_g$ alors le robot effectue un rotation sur lui-même. L'utilisation de ces deux seuls modes de locomotion, bien que limitée, permet de découpler les mouvements et de fournir une solution simple pour amener le robot d'une posture à une autre. C'est sans doute là une des raisons du succès de ce type de robots. Pour élaborer une stratégie plus fine de déplacement, il est cependant intéressant de savoir comment la posture du robot est reliée à la commande de ses roues.

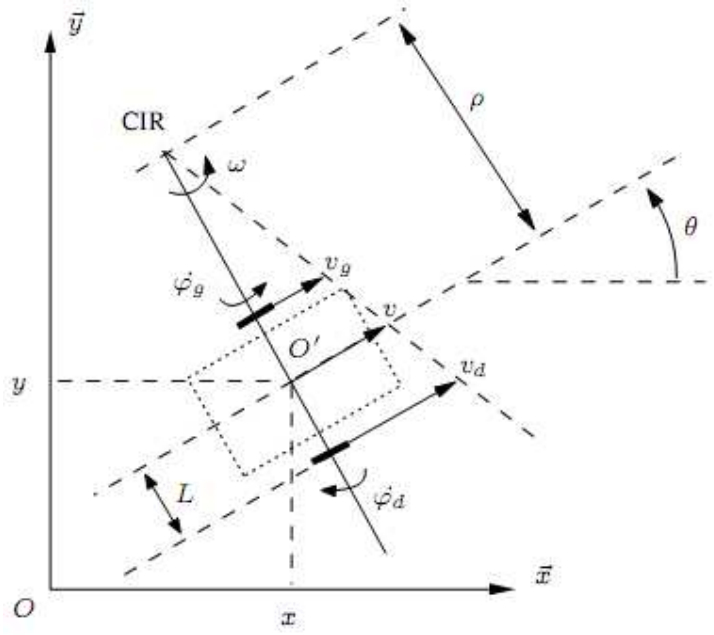

Figure 4.11 Centre instantané de rotation d'un robot de type unicycle

c) Choix de la commande En ce qui concerne la commande, si l'on se contente de traiter le cas cinématique, on peut considérer que celle-ci est donnée, au plus bas niveau, par les vitesses de rotation des roues. Ceci étant, on préfère généralement exprimer cette commande par la vitesse longitudinale du robot, notée v (en $\acute{O}$) et sa vitesse de rotation $\dot{\theta}$ (autour de $\acute{O}$). Il y a en effet équivalence entre les deux représentations. D'une part, on a :

$$v = \frac{v_g + v_d}{2} = \frac{r(\dot{\varphi}_g - \dot{\varphi}_d)}{2} \qquad (\mathbf{4.10})$$

D'autre part, la vitesse de rotation du robot est égale à la vitesse de rotation autour du CIR **[DUD 00]** :

$$\omega = \dot{\theta} = -\frac{r(\dot{\varphi}_g + \dot{\varphi}_d)}{2L} \qquad (\mathbf{4.11})$$

Conformément à l'équation **(4.10)** et $(\mathbf{4.11})$**.** On montre que ces relations sont parfaitement inversibles et qu'il y a ainsi équivalence entre les couples $(\dot{\varphi}_g, \dot{\varphi}_d)$ et (v, ω). Désormais, on utilise plutôt ce dernier couple de grandeurs, plus parlantes, quitte à calculer ensuite les angles ou vitesses de consigne des asservissements des roues.

d) Modèle cinématique Relier la dérivée de la posture à la commande $u = (v\ \omega)^T$ est facile. Une simple considération géométrique donne :

$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \omega$$

Ce qui s'écrit, sous forme matricielle :

$$\dot{\varepsilon} = \mathrm{C(q)u} \qquad (4.12)$$

Avec :

$$C(q) = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix}$$

Ce modèle est appelé modèle cinématique en posture du robot **[CAM 96].**

3.3.3 Robots mobiles de type tricycle et de type voiture

Ces robots partagent des propriétés cinématiques proches, raison pour laquelle on les regroupe ici.

a) Description

Considérons tout d'abord le cas du tricycle, représenté par la **figure 4.12.** Ce robot est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. De ce point de vue, il est donc très proche d'une voiture. C'est d'ailleurs pour cela que l'on étudie le tricycle, l'intérêt pratique de ce type de robot (peu stable !) restant limité.

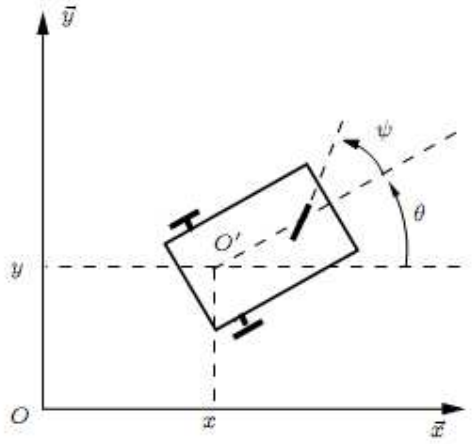

Figure 4.12 Robot mobile de type tricycle

Le cas du robot de type voiture est très similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comporte deux roues au lieu d'une. Cela va de soit, on rencontre beaucoup plus souvent ce type de systèmes. On parle de robot dés lors que la voiture considérée est autonome **[KAN 86**], donc sans chauffeur ni télépilotage. Il s'agit là d'un des grands défis issus de la robotique mobile.

b) Modélisation

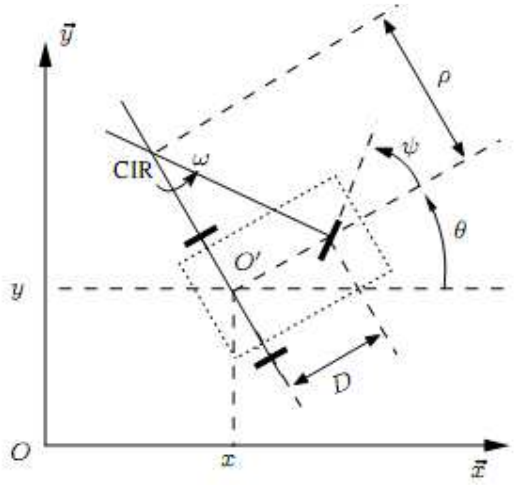

Figure 4.13 Un robot mobile de type tricycle et son CIR

Considérons tout d'abord le tricycle. Le CIR du robot se situe à la rencontre des axes des roues fixes et de la roue orientable, comme cela est représenté à **la figure 4.13**. On peut déterminer ρ de manière géométrique à partir de l'angle d'orientation de la roue avant et ω à partir de la vitesse linéaire v du véhicule (vitesse en $\acute{O}$) et de ρ :

$$\rho = \frac{D}{\tan\varphi} \ , \qquad (\mathbf{4.13})$$

$$\omega = \frac{\mathrm{v}}{D} \tan\varphi \ , \qquad (\mathbf{4.14})$$

Ce type de robot peut se diriger en ligne droite pour φ = 0 et théoriquement tourner autour du point $\acute{O}$ (on pourrait dire sur place) pour φ = π/2. Néanmoins, le rayon de braquage de la roue orientable, généralement limité, impose le plus souvent des valeurs de φ telles que $-\pi/2 < \varphi < \pi/2$, interdisant cette rotation du robot sur lui même.

L'écriture des contraintes sur chacune des roues et un raisonnement similaire à celui suivi dans le cas de l'unicycle permettent de déterminer les modèles cinématiques des robots de type tricycle. Toutefois, par un simple raisonnement géométrique, on établit les équations **[BER 09]**:

$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \frac{v}{D}\tan\varphi$$

$$\dot{\varphi} = \eta$$

Où u = (v η)T est le vecteur de commande cinématique, η représentant la vitesse d'orientation imposée à la roue orientable. Ces équations sont celles du modèle cinématique en configuration simplifiée, la configuration associée q = (x y θ φ)^T ne d'ecrivant pas les rotations propres des différentes roues.

Comme on l'a vu précédemment, l'existence d'un CIR unique impose que les axes des roues du robot soient concourants. Dans le cas du robot de type voiture, cela impose aux roues du train avant de n'avoir pas la même orientation, comme illustré à la **figure 4.14**. Ainsi, les roues avant d'un robot de type voiture (et a fortiori d'une voiture) ne sont pas parallèles. Le roulement idéal, assurant que le CIR est bien unique, est réalisé sur une voiture par un système de braquage différentiel (dit d'Ackerman). Par ailleurs, les trajectoires des roues n'ayant pas un même rayon de courbure, leurs vitesses sont également différentes (et liées évidemment).

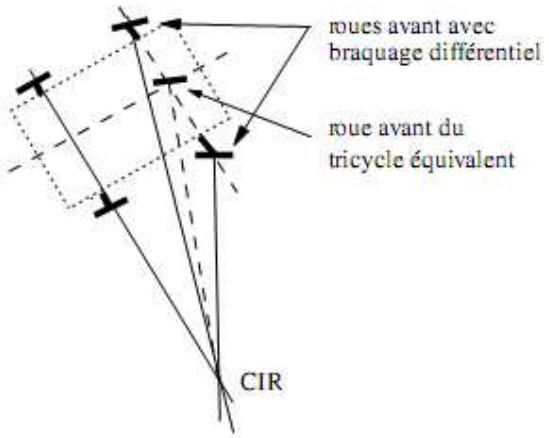

Figure 4.14 Un robot mobile de type voiture et son CIR

L'équivalence entre tricycle et voiture est facile à montrer. Il suffit pour cela de figurer une roue virtuelle qui transformerait un robot de type voiture en tricycle en plaçant la roue orientable du tricycle au centre de l'axe des roues avant de la voiture, orientée de sorte que le CIR reste inchangé, conformément à la **figure 4.14 [BER 09].**

3.3.4 Robots mobiles omnidirectionnels

a) Description

Un robot mobile est dit omnidirectionnel si l'on peut agir indépendamment sur les vitesses : vitesse de translation selon les axes x et y et vitesse de rotation autour de **z**.D'un point de vue cinématique on montre que cela n'est pas possible avec des roues fixes ou des roues centrées orientables **[CAM 96].** On peut en revanche réaliser un robot omnidirectionnel en ayant recours à un ensemble de trois roues décentrées orientables ou de trois roues suédoises disposées aux sommets d'un triangle équilatéral (**figure 4.15**). Du point de vue de la transmission du mouvement, ceci ne va pas sans poser de problème.

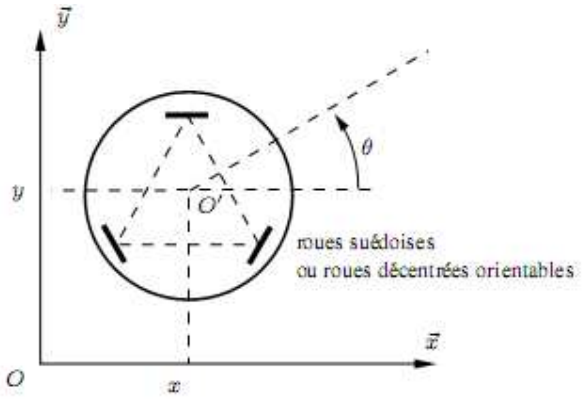

Figure 4.15 Représentation d'un robot mobile omnidirectionnel

b) Modélisation

Dans ce cas on peut considérer qu'il est possible de commander directement la posture et le modèle cinématique en posture est donc **[BER 09]**:

$$\dot{x} = u_1$$

$$\dot{y} = u_2$$

$$\dot{\theta} = u_3$$

où $u = (u_1\ u_2\ u_3)^T$ représente le vecteur de commande. On choisit ainsi généralement ce type de robot pour se dispenser des problèmes de planification et de commande liés à la non-holonomie. L'avantage d'une cinématique extrêmement simple est cependant à mettre en balance avec les inconvénients liés à une localisation odométrique déficiente et à une plus grande complexité mécanique, généralement responsable d'un surcoût.

3.4 Propriétés du modèle cinématique d'un robot

3.4.1 Représentation d'état [BER 09]

De la modélisation que nous avons exposée tout au cours de ce chapitre il résulte, dans tous les cas, un modèle dynamique (au sens de l'Automatique) sous la forme $\dot{q} = B(q)u$.

Si l'on considère que la configuration (réduite) q fait office de variable d'état du système on a :

$$\dot{q} = B(q)u \qquad (\mathbf{4.15})$$

Avec B(q) de dimension n×m. Il s'agit d'une représentation non linéaire sans terme de dérive, en comparaison avec la représentation d'état classique d'un système linéaire invariant $\dot{x}$ = Ax + Bu.

C'est le cas des robots mobiles de type unicycle ou voiture, qui par ailleurs font partie d'une classe de système non linéaires particuliers dits systèmes chainés, comme on le précisera au paragraphe consacré à la commande des robots.

3.4.2 Commandabilité des robots mobiles à roues

Pour qu'un robot mobile soit utile (ou utilisable) il faut en premier lieu s'assurer de sa commandabilité. Cette propriété signifie qu'il existe toujours une loi de commande u(t) amenant le robot d'un état initial à un état final quelconque.

Pour caractériser la commandabilité d'un système non linéaire, on peut tout d'abord se demander si son modèle linéarisé autour de tout point d'équilibre est commandable. En x_0 quelconque, le modèle linéarisé issu de **(4.15)** s'écrit :

$$\dot{x} = B(x_0)u$$

avec $B(x_0) \in R^{n\times m}$. Pour les robots de type unicycle et voiture, la non-holonomie va de pair avec une forme de sous-actionnement qui se traduit par le fait que m < n. Appliquée à un tel système, la condition de rang (critère de commandabilité de Kalman) :

$$Rang(B, AB, \dots, A^{n-1}B) = n$$

Se résume à :

$$Rang\big(B(x_0)\big) = n$$

Cette condition n'est jamais remplie puisque m < n. Le modèle linéarisé du système n'est donc pas commandable autour d'un point d'équilibre quelconque.

Il faut alors, pour statuer sur la commandabilité du système, utiliser le théorème de Chow. On donne ici (sans preuve) son interprétation pour un système dont le modèle cinématique est donnée par **(4.15).**

Théorème Commandabilité d'un robot mobile non holonome [BER 09]

On note B(x) = (b1(x) b2(x) . . . bm(x)) la matrice du modèle cinématique **(4.15),** de dimension n × m.

Un robot mobile est commandable si les colonnes de B(x) et leurs crochets de Lie successifs forment un ensemble de n colonnes indépendantes.

On considère le cas de l'unicycle. A partir du leur modèle cinématique et les crochets de Lie et en utilisant le théorème de Chow on montre que le modèle cinématique de l'unicycle est commandable. On montrerait de même que celui de la voiture l'est aussi. La conséquence de cette propriété est l'existence de retours d'états pour commander le système, comme on le verra au paragraphe suivant.

4 Stabilisation de trajectoire pour un robot mobile

Le problème de stabilisation consiste à calculer pour le robot, la trajectoire admissible, joignant la configuration initiale à la configuration finale (l'origine) avec des commandes initiales et finales. La tâche du robot est définie comme l'atteinte de la destination finale.

Pour chaque robot, l'objectif est la synthèse d'une commande en boucle fermée assurant sa stabilisation. Cela veut dire, définir un ensemble fini de trajectoires permettant de joindre la position d'origine à partir de n'importe quelle configuration initiale.

Dans **[SAM 08]** en remarque que la stabilisation par retour d'état discontinu invariant en forme chaînée est plus performante par rapport aux techniques de feedback linearization (linéarisation par bouclage). Pour cela nous présentons dans cette loi de commande pour des systèmes non-holonomes, nous appliquons cette commande à une forme simplifiée et non au modèle cinématique de robot mobile. Cette forme est appelée la forme chaînée **[SAM 95].**

Nous donnons ci-dessous une brève introduction sur la forme chaînée d'ordre « n » pour ensuite présenter la synthèse de la loi de commande stabilisante correspondante.

4.1 La forme chaînée

C'est l'une des manières qui permet de mettre un système sous une certaine forme canonique qui facilite l'analyse et le calcul des contrôles.

L'allure générale de cette forme est donnée par **[SAM 08]** :

$$\begin{cases} \dot{x_1} = u_1 \\ \dot{x_2} = u_2 \\ \dot{x_3} = x_2 u_1 \\ \quad \vdots \\ \dot{x_n} = x_{n-} u_1 \end{cases} \qquad (4.16)$$

Où $x = (x_1 \quad x_2 \quad \cdots \quad x_n)^T$ représente le vecteur d'état et $(u_1 \quad u_2)^T$ indique le vecteur d'entrée du système **(4.16).**

Cette forme est appelée forme **chaînée simple (2, n)** où 2 représente le nombre de contrôle et n le nombre d'états.

4.2 Algorithme de génération de trajectoire stabilisante

L'objectif de la commande stabilisante est de ramener le robot à une position d'équilibre à partir d'une position quelconque. La trajectoire dans ce cas n'est pas définie au préalable. C'est le type de commande appliquée qui désignera la trajectoire que doit suive le robot pour atteindre la position d'équilibre souhaitée.

Soit un système non-holonome dont la forme chaînée correspondante est donnée par **le système (4.16).** Stabiliser le système non-holonome revient à stabiliser sa forme chaînée correspondante.

Les deux commandes peuvent s'écrire **[AST 96]** :

$$\begin{cases} u_1(x) = v_1(x) \\ u_2(x) = v_2(x) + w_2(x) \end{cases} \qquad (4.17)$$

Pour calculer les différents termes de $u_1(x)\ et\ \ u_2(x)$ nous procédons de la manière suivante :

Posons au départ que $w_2(x) = 0$ et le retour d'état linéaire $(v_1(x) \quad v_2(x))^T$ tel que :

$$u_1(x) = v_1(x) = -k_1 x_1 \qquad (4.18)$$

$$u_2(x) = v_2(x) = -k_2 x_1 - k_3 x_2 \qquad (4.19)$$

Où k_i sont des réels et $k_1\ et\ k_3$ sont strictement positifs et $k_1 \neq k_3$.

Le système en boucle fermée peut être intégré, étape par étape, pour obtenir :

$$x_1(t) = x_{10}\exp(-k_1 t)$$
$$x_2(t) = \left(x_{20} - \frac{k_2 x_{10}}{k_a}\right)\exp(-k_3 t) + \frac{k_2 x_{10}}{k_a}\exp(-k_1 t)$$
$$x_3(t) = \frac{k_1}{k_b}x_{10}\left(x_{20} - \frac{k_2 x_{10}}{k_a}\right)\exp(-k_b t) + \frac{k_2}{2k_a}{x_{10}}^2\exp(-2k_1 t) + S_3(x_0)$$
$$x_4(t) = \frac{{k_1}^2{x_{10}}^2}{k_b(k_1+k_b)}\left(x_{20} - \frac{k_2 x_{10}}{k_a}\right)\exp(-(k_1+k_b)t) + \frac{k_2}{6k_a}{x_{10}}^3\exp(-3k_1 t) + x_{10}S_3(x_0)\exp(-k_1 t) + S_4(x_0)$$

……..

$$x_n(t) = \frac{{k_1}^{n-2}{x_{10}}^{n-2}}{k_b(k_1+k_b)\dots(k_b+(n-3)k_1)}\left(x_{20} - \frac{k_2 x_{10}}{k_a}\right)\exp(-((n-3)k_1+k_b)t) + \frac{k_2}{(n-1)!\,k_a}{x_{10}}^{n-1}\exp(-(n-1)k_1 t) + \sum_{i=1}^{n-3}\frac{{x_{10}}^i}{i!}S_{n-i}(x_0)\exp(-ik_1 t) + S_n(x_0) \qquad (4.20)$$

Avec $k_a = k_1 - k_3$, $k_b = k_1 + k_3$ et $S_3(x_0)$, $S_4(x_0)$, $\dots S_n(x_0)$ sont des constantes d'intégrations qui peuvent être déterminées à t=0, comme une fonction des conditions initiales $x_{10}, x_{20}, \dots x_{n0}$.

A partir de **(4.20),** nous constatons que (x_1,x_2, x_3, x_4 … x_n) tendent vers ($0, 0, S_3(x_0)$, $S_4(x_0)$, …$S_n(x_0)$) quand t tend vers l'infini.

Dans ce cas, si nous prenons des conditions initiales tel que $S_j(x_0)$ =0 ($3 \leq j \leq n$), tous les états tendent, alors vers l'origine.

En résumé la stabilisation de **la forme chaînée (4.16)** est équivalente à la stabilisation de ($0, 0, S_3(x_0)$, $S_4(x_0)$, …$S_n(x_0)$) en prenant en compte le paramètre $w_2(x)$.

Dans notre cadre d'étude $w_2(x)$ est calculé de la façon suivante :

$$w_2(x) = C^T Q(x_1) S \qquad (4.21)$$

Avec :

$$C = \begin{bmatrix} C_3 \\ C_4 \\ \vdots \\ C_n \end{bmatrix}, Q(x_1) = \begin{bmatrix} 1/x_1 & 0 & \cdot & 0 \\ 0 & {1/x_1}^2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & 0 & {1/x_1}^{n-2} \end{bmatrix}, S = \begin{bmatrix} S_3(x_0) \\ S_4(x_0) \\ \vdots \\ S_n(x_0) \end{bmatrix}.$$

Le vecteur C est choisi de telle sorte que la valeur de la matrice $(A + B.C)^T$ possède des parties réelles négatives.

Les expressions de A et B sont données ci-dessous :

$$A = \begin{bmatrix} k_1 & 0 & \cdot & 0 \\ 0 & 2k_1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & 0 & (n-2)k_1 \end{bmatrix}, B = \begin{bmatrix} b_3 \\ b_4 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \frac{-k_1}{k_b} \\ \frac{k_1}{k_1 + k_b} \\ \vdots \\ \frac{(-1)^n k_1}{(n-3)!\,((n-3)k_1 + k_b)} \end{bmatrix}$$

4.3 Synthèse d'une loi de commande pour un robot mobile

Le robot mobile considéré est un robot de type unicycle avec deux roues avant motrice et une roue arrière stabilisante. La commande de ce type de robot est établie par l'intermédiaire d'une vitesse linéaire d'une pointe M notée « v » et d'une vitesse angulaire dans la direction des roues avant notée « w ».

Le robot est supposé dans un environnement de roulement sans glissement dans un sol horizontal. La configuration de ce type de robot est décrite par le vecteur (x, y, θ), où (x, y) sont les coordonnées du point M localisée à la mi-distance des deux roues avant, θ représente l'orientation du véhicule, entre l'axe du robot et l'axe des abscisses x.

Le modèle cinématique du robot est donné par l'expression suivante :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w \qquad (4.22)$$

Ici l'objectif est de commander le robot afin de le ramener à une position d'équilibre en suivant une trajectoire donnée.

Pour appliquer la technique de la commande stabilisante au robot concerné, nous transformons la forme décrite dans l'équation **(4.22)** en forme chaînée d'ordre 3. La forme chaînée, représentant le modèle du robot unicycle, est donnée par l'équation suivante :

$$\begin{cases} \dot{x_1} = u_1 \\ \dot{x_2} = u_2 \\ \dot{x_3} = x_2 u_1 \end{cases} \qquad (4.23)$$

L'expression **(4.23)** est obtenue en effectuant le changement de variable suivant :

$$\begin{cases} x_1 = x \\ x_2 = \tan\theta \\ x_3 = y \end{cases}$$

La relation donnant les entrées de commande « v, w » en fonction des nouveaux commandes «u_1, u_2 » est la suivante :

$$\begin{cases} v = \dfrac{u_1}{\cos\theta} \\ w = u_2 \cos^2\theta \end{cases} \qquad (4.24)$$

Pour ramener le robot à une position d'équilibre, nous allons concevoir une commande $(u_1(x), u_2(x))^T$ avec un retour d'état discontinu invariant.

Calcul les termes de u_1 et u_2

Les expressions des commandes $u_1(x)$ et $u_2(x)$ sont données comme suit :

$$u_1(x) = v_1(x) \qquad (4.25)$$

$$u_2(x) = v_2(x) + w_2(x) \quad (4.26)$$

D'abord, posons $w_2(x) = 0$ et considérons le retour d'état linéaire $(v_1(x) \quad v_2(x))^T$ tels que :

$$u_1(x) = -k_1 x_1 \qquad (4.27)$$
$$u_2(x) = -k_2 x_1 - k_3 x_2 \quad (4.28)$$

Où k_i sont des réels et $k_1 \; et \; k_3$ sont strictement positifs et $k_1 \neq k_3$

Le système en boucle fermée peut être intégré pour obtenir les états suivants **[SAM 08]** :

$$x_1(t) = x_{10}\exp(-k_1 t)$$
$$x_2(t) = \left(x_{20} - \frac{k_2 x_{10}}{k_a}\right)\exp(-k_3 t) + \frac{k_2 x_{10}}{k_a}\exp(-k_1 t) \qquad (4.29)$$
$$x_3(t) = \frac{k_1}{k_b}x_{10}\left(x_{20} - \frac{k_2 x_{10}}{k_a}\right)\exp(-k_b t) + \frac{k_2}{2k_a}{x_{10}}^2\exp(-2k_1 t) + S_3(x_0)$$

Avec$k_a = k_1 - k_3$, $k_b = k_1 + k_3$ et $S_3(x_0)$ est un constante d'intégration qui peut être déterminée, à t=0, comme une fonction des conditions initiales x_{10}, x_{20}.

A partir de **(4.29),** on observe que (x_1, x_2, x_3) tendent vers $(0, 0, S_3(x_0))$ quand t tend vers l'infini. Alors, si nous prenons les conditions initiales de telle sorte que $S_3(x_0) = 0$, tous les états tendent vers l'origine.

Prenons $S_3(x_0) = S_3$, de tel sorte que les conditions initiales sont satisfaites (remplacer t=0 dans **(4.29)** et déterminer $S_3(x_0)$ et ensuite substituer x_0 par x). Nous obtenons l'expression suivante :

$$S_3(x) = x_3 - \frac{k_1}{k_b}x_1 x_2 + \frac{k_2}{2k_a}{x_1}^2 \qquad (4.30)$$

En résumé la stabilisation de la forme chaînée **(4.23)** est équivalent à la stabilisation de $(0, 0, S_3(x_0))$ en prenant en compte le paramètre $w_2(x)$ qui est calculé de la façon suivante :

$$w_2(x) = C^T Q(x_1) S \qquad (4.31)$$

Avec :

$C = C_3, Q(x_1) = \frac{1}{x_1}$, et $S = S_3$.

Le vecteur C est choisi en fonction du signe de $(A + B.C)^T$. Ce dernier doit posséder des parties réelles négatives.

Dans ce cas nous avons alors :

$$A = k_1$$

$$B = \frac{-k_1}{k_b}$$

$$(A + B.C)^T = k_1 - \frac{k_1}{k_b}C = k_1(1 - \frac{C}{k_b})$$

La valeur de $(A + B.C)^T$ possède des parties réelles négatives, c'est équivalent à dire que :

$$k_1\left(1 - \frac{C}{k_b}\right) < 0 \Rightarrow \left(1 - \frac{C}{k_b}\right) < 0 \Rightarrow C > k_b \Rightarrow C > k_1 + k_3$$

Au final, l'expression de contrôleur $(u_1(x), u_2(x))^T$ qui stabilise notre robot est donné par :

$$u_1(x) = -k_1 x_1 \qquad (4.32)$$

$$u_2(x) = -k_2 x_1 - k_3 x_2 - \frac{C_3 S_3}{x_1} \qquad (4.33)$$

5. Conclusions

Dans ce chapitre, nous avons rappelé le principe de la commande référencée vision d'un robot mobile non-holonome muni d'une caméra. Nous avons alors décrit brièvement le système robotique considéré, et établi les modèles nécessaires pour exprimer le lien entre le mouvement de la caméra et la variation des indices visuels. Nous avons ensuite présenté les différents types de robots mobiles à roues les plus utilisés. Cette variété réside dans leur mode de locomotion qui dépend du type et de la disposition des roues utilisées. Une étude cinématique spécifique était donc nécessaire pour chaque type de robot avant toute étape de développement.

Nous avons montré aussi la différence entre un système holonome et un système non-holonome. Nous avons observé que la majorité des robots mobiles sont des systèmes non-holonomes excepté le cas des robots omnidirectionnels.

Ensuite une méthode de commande à été synthétisée afin de garantir la stabilisation d'un robot mobile de type unicycle. Cette méthode consiste à ajouter un terme discontinu à une commande par retour d'état linéaire qui stabilise le robot modélisé sous la forme chaînée.

Implémentation de l'approche du V-disparité

1. Introduction

Ce chapitre est consacré au portage de l'algorithme de la stéréovision sur une plate-forme matérielle afin de répondre aux contraintes temps réel qui nous sont imposées. Nous présentons dans un premier temps les différentes étapes de l'approche de V-Disparité, par la suite l'implémentation de chaque étape de la stéréovision est explicitée et évaluée en termes d'architecture, de ressources consommées et de performances temps réel.

2. L'organigramme général de l'approche V-Disparité

Cette approche est basée sur la construction et l'analyse de l'image « v-disparité », qui fournit une bonne représentation géométrique de la scène tridimensionnelle.

L'organigramme général de l'implémentation hardware de la v-disparité peut être donné par la figure suivante :

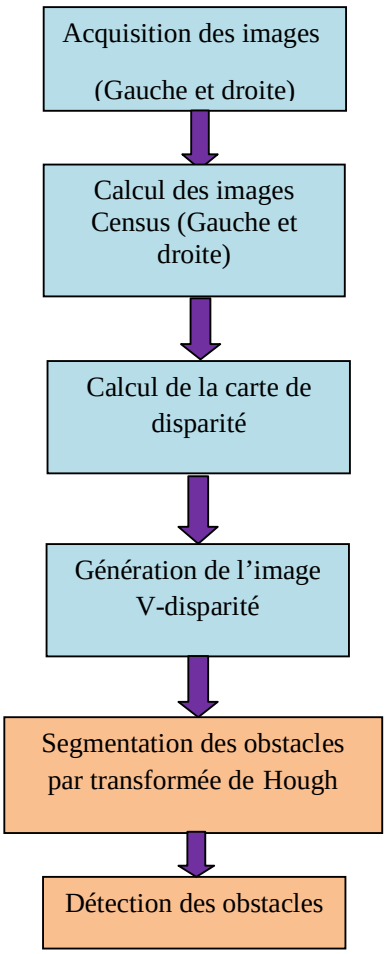

Figure 5.1 : L'organigramme de l'approche V-disparité

3. L'architecture d'implémentation de la V-Disparité sur FPGA

L'architecture globale pour l'implantation de l'algorithme comprenant cinq étapes : l'acquisition des images, le calcul de la transformée de Census, la mesure de l'appariement (le calcul des scores) et la recherche des disparités et enfin la génération de l'image de V-Disparité.

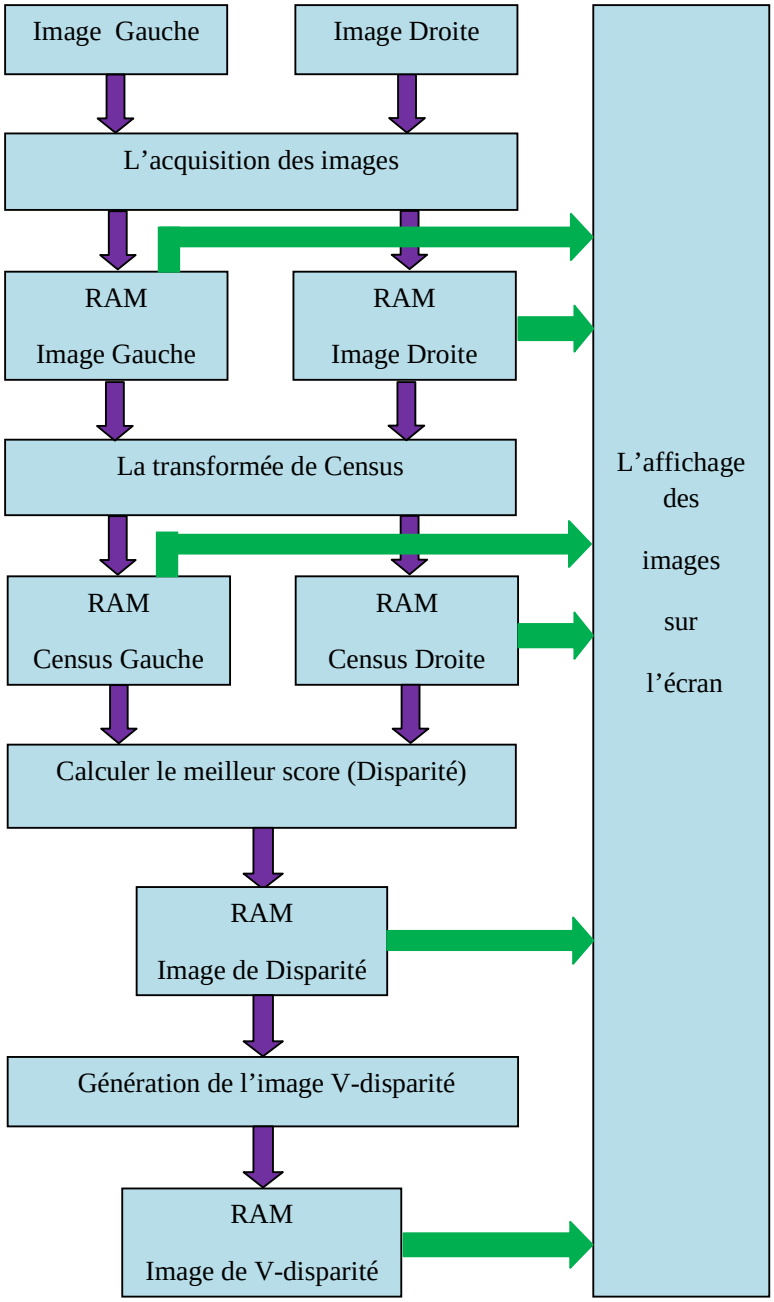

Figure 5.2 Schéma synoptique de l'architecture

A chaque étape nous affichons les images résultantes par une procédure d'affichage des images mémoires sur l'écran de la carte RC203E.

Lors de la proposition de cette architecture, on a posé les hypothèses suivantes :

- Les images stéréoscopiques sont rectifiées : la recherche du correspondant se fera sur la même ligne. Cette hypothèse est justifiée par la position des deux camera dans l'organe stéréoscopiques (**figure 1.4 :** Modélisation du capteur stéréoscopique).
- La taille des images utilisées est : 480X640 pour avoir la même résolution entre la camera et l'écran de la carte.
- La disparité maximale est préalablement connue. L'étape de simulation nous permet d'avoir une idée de l'ordre de grandeur de cette valeur. Cette connaissance à priori permet de rechercher le correspondant dans un intervalle connu. Le nombre des codes Census à calculer est donc connu. On pourra aussi déduire directement la taille de l'image de la V-Disparité. Dans notre cas la disparité maximale est 64.
- La transformée de Census est calculée en utilisant des fenêtres 3X3. Donc les codes Census calculés ont une taille de 8 bits, donc les images Census calculées sont des images a niveau de gris.

3.1 L'acquisition des images

L'acquisition du signal vidéo en utilisant la carte RC203E de CELOXICA est assurée en empruntant les étapes suivantes :

- La capture du signal vidéo par une camera dotée de trois pins : la terre, l'alimentation et la pin du signal vidéo.
- La canalisation du signal capté vers un processeur multi tâches pour avoir un signal vidéo PAL.
- La réception du signal numérique (Standard PAL entrelacé, codé en Y-Cb-Cr) par le composant FPGA. Le signal reçu doit être convertis en signal RGB 24 bits. Les signaux de synchronisation doivent être transformés à des adresses mémoires.
- L'arrangement des pixels RGB 24 bits dans la mémoire RAM est suivant leurs adresses.

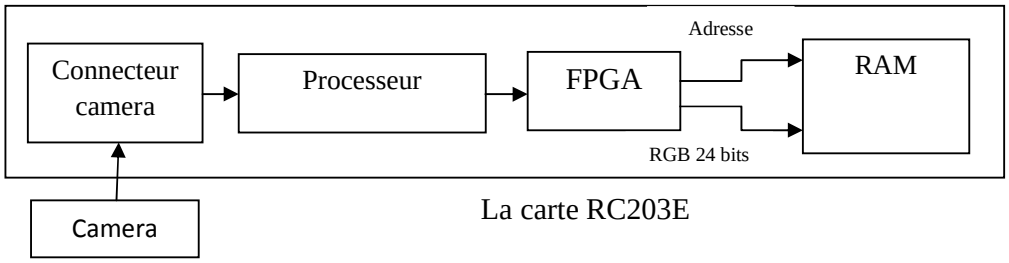

Figure 5.3 La chaîne d'acquisition d'un signal vidéo par la carte RC203E

3.2 L'affichage des images

L'affichage des images mémoires sur l'écran de la carte RC203E ce fait en suivant les étapes suivantes :

- Le composant FPGA charge les pixels RGB 24 bits et leurs adresses à partir de la RAM.
- Le composant FPGA transforme les adresses mémoires aux signaux de synchronisation.
- La réception du signal composé des 24 bits RGB et 6 bits synchronisation par un DAC. Les signaux RGB seront transformés en signaux analogiques de standard VGA. A la sortie du DAC, cinq signaux analogiques seront récupérés : 1 signal pour chaque couleur et deux signaux pour les synchronisations horizontale et verticale.
- La canalisation des 5 signaux analogiques via le connecteur VGA afin de visualiser les images.

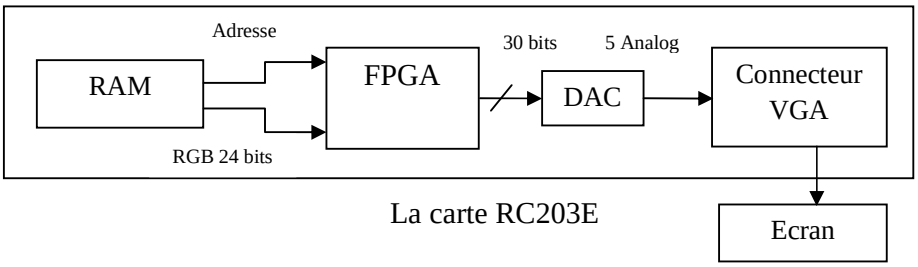

Figure 5.4 La chaîne d'affichage des images par la carte RC203E

3.3 Implémentation de la transformée de Census

L'image Census est obtenue en scannant l'image par une fenêtre dont la taille est un élément essentiel pour faciliter l'appariement entre *les* images gauche et droite. Dans chaque fenêtre de l'image, la valeur de l'intensité du pixel de référence (le pixel central de la fenêtre) est comparée avec celles des pixels voisins. Chaque pixel voisin est représenté par un bit dans le code Census.

Les pixels de l'image moyennée sont stockés au rythme de l'horloge pixel dans un buffer circulaire composé de trois mémoires double-port **(figure 5.5).** Les sorties de ces mémoires sont connectées à leur tour à une matrice composée de trois lignes et trois colonnes de registres à décalage (9 registres à décalage au total). Le rôle de cette matrice est de stocker les valeurs du pixel central et de ses pixels voisins pour la fenêtre considérée. Ces valeurs sont ensuite traitées en temps réel pour fournir un flot de codes Census sur 8 bits.

Les pixels RGB stockés dans la RAM sont convertis en pixels 8 bits par la formule suivante :

Pixel=0.11xR+0.59xG+0.3xB

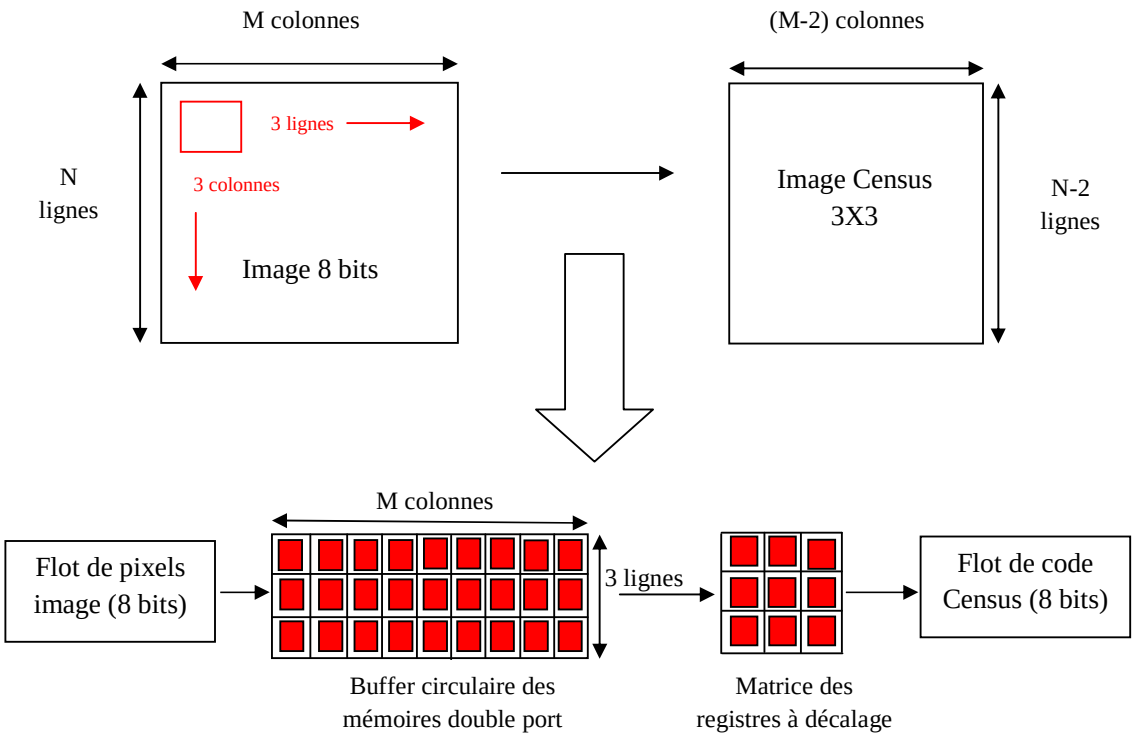

Figure 5.5 Principe de la transformation Census à partir d'une fenêtre 3 x 3

Les codes Census sont calculés à partir de matrice des registres à décalage par une simple comparaison entre le pixel central et ces pixels voisins.

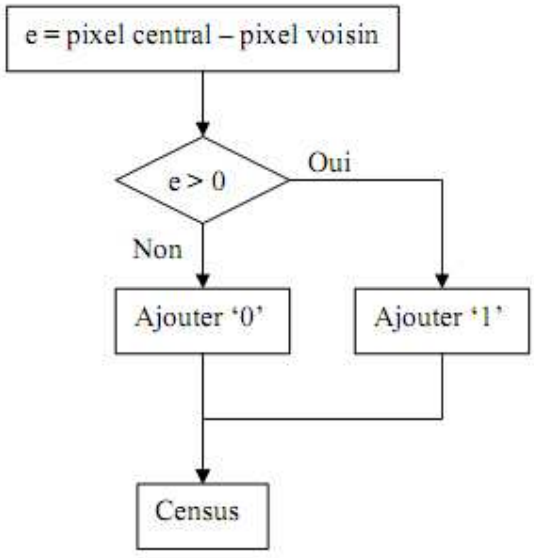

Figure 5.6 : Algorithme de calcul de la transformée de Census.

Donc le code Census est formé de 8 comparateurs fonctionnant en parallèle comme le montre la figure suivante :

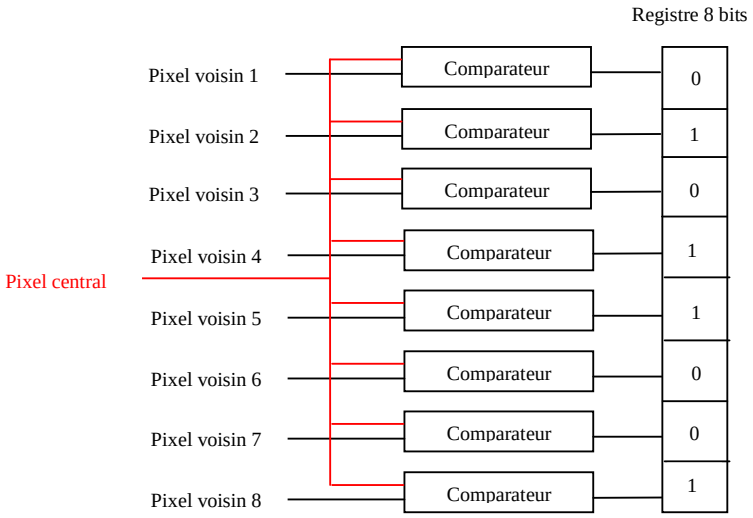

Figure 5.7 Calculateur de code Census

3.4 Mesure de l'appariement et recherche des disparités

A partir des deux images Census (gauche et droite), on cherche pour chaque pixel gauche son meilleur correspondant dans une chaîne de pixels de l'image droite. Pour cela on calcule dans un premier temps pour chaque pixel candidat de l'image droite, le score associé à l'appariement : celui-ci correspond au nombre de bits identiques entre le pixel gauche et le pixel de droite concerné.

Dans un deuxième temps on effectue une recherche du meilleur score parmi les scores issus des pixels candidats.

Pour réaliser ces opérations, le flot des pixels de l'image droite traverse une chaîne de registres à décalage (Dmax registres) préalablement initialisée à 0 **(figure 5.8)**. Les scores sont calculés en appliquant simultanément la fonction XNOR entre le pixel gauche motif et les sorties de la chaîne de registres. On obtient les scores en comptabilisant, à partir de fonctions combinatoires et pour chaque pixel candidat, le nombre de comparaisons réussies lors de cette opération. Les Dmax scores obtenus sont ensuite appliqués à un bloc de tri fonctionnant à la fréquence pixel et dont le rôle est d'extraire le meilleur score et son rang.

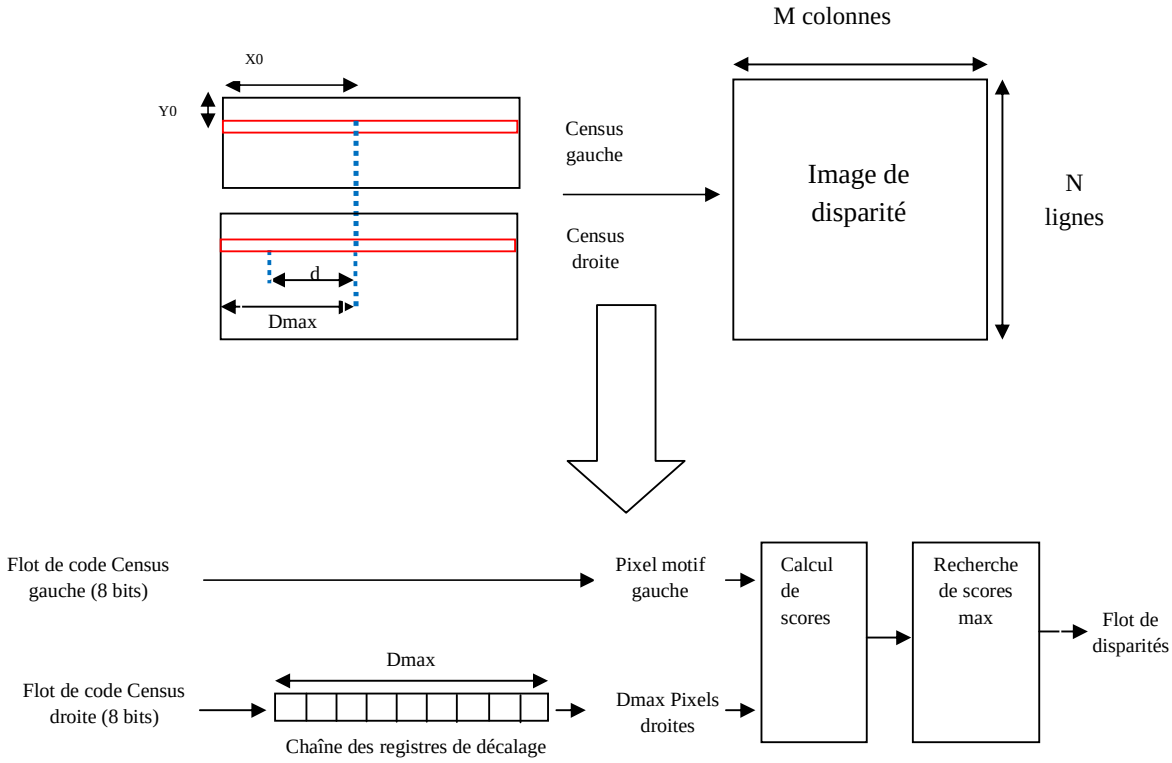

Figure 5.8 Etapes pour la recherche des disparités

Le calcul des scores est en fait un calcul du poids de correspondance entre le Census du pixel motif et le Census des pixels candidats. Ce calcul se fait en appliquant d'abord la fonction logique XNOR entre les différents Census (pixel motif et tous les pixels candidats) en même temps. Le degré d'appariement entre le motif et les candidats est représenté par le nombre de '1' présents dans chaque chaîne à la sortie de la fonction XNOR. Ce nombre de « 1 » est comptabilisé pour fournir un « score » ou « poids de correspondance » représentatif du niveau de ressemblance entre le motif d'origine et les motifs candidats.

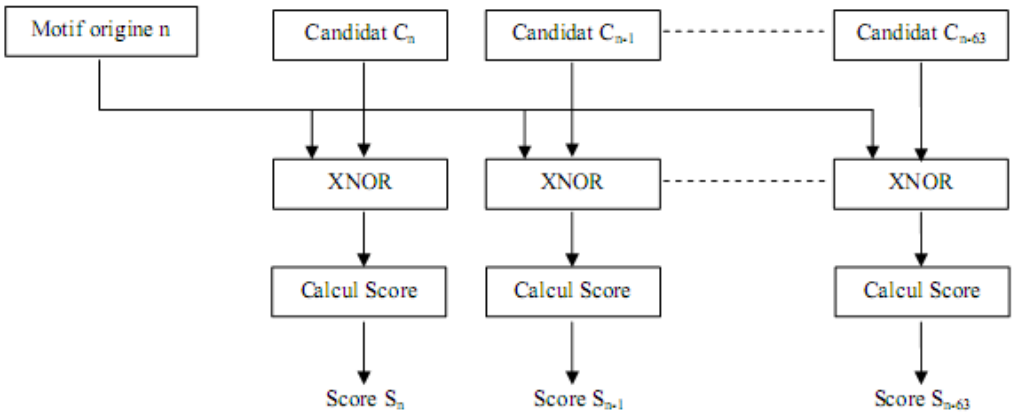

Figure 5.9 : Le calcul des scores.

Le principe de l'extraction du score maximal est simple, ce dernier est le maximum d'un ensemble de scores. On les compare 2 à 2. Le schéma est combinatoire, il est rapide. On récupère le maximum immédiatement après l'application des entrées.

Après l'extraction du score maximal, le calcul de la disparité consiste à déterminer le rang de ce score. Pour se faire, on compare le score maximal avec tous les scores. Le rang de la première valeur qui coïncide avec le score maximal représente la valeur de la disparité.

3.5 Génération de l'image V-Disparité

Une fois la carte de disparité calculée, l'image « v-disparité » est construite en accumulant pour chaque ligne image tous les pixels de disparité connue.

Ainsi, pour la ligne image i, l'abscisse u_m du point M dans l'image « v-disparité » correspond à la disparité de ce point Δ_M et la valeur de son niveau de gris correspond au nombre de points de même disparité sur la ligne i. Le processus d'accumulation ainsi mis en œuvre est essentiel et confère à la méthode ses propriétés de robustesse.

La figure suivante présente l'architecture proposée pour l'implémentation de l'étape de génération de V-Disparité sur la carte RC203E.

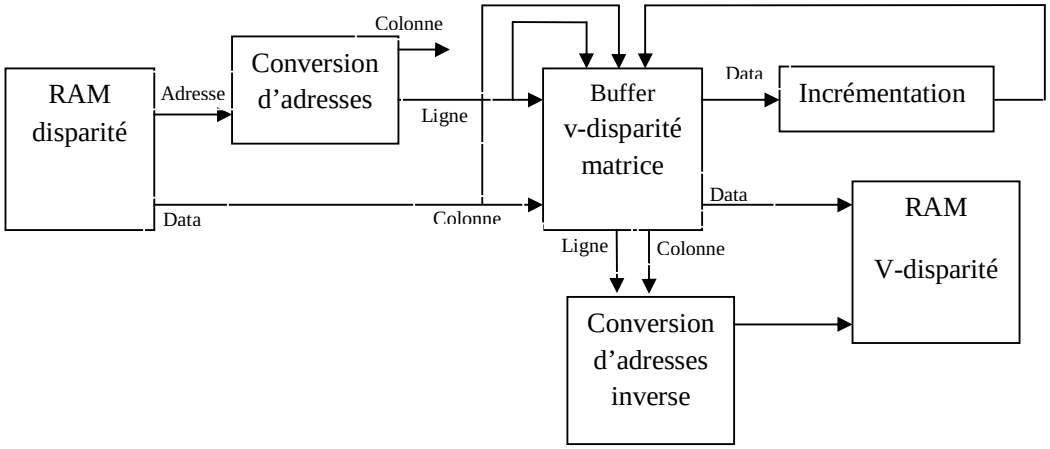

Figure 5.10 Implémentation du calcul de l'image V-Disparité

Dans cette architecture :

- Le buffer c'est une mémoire sous forme matricielle, ce type de buffer est disponible sur la carte RC203E.
- Les cases de la matrice du buffer sont initialement nulles.
- La procédure de conversion d'adresses permet de convertir l'adresse mémoire de pixel au numéro de ligne et numéro de colonne de pixel dans l'image.
- La procédure de conversion d'adresses inverse permet de convertir le numéro de ligne et le numéro de colonne de pixel à l'adresse mémoire de pixel.

La procédure de calcul de l'image de V-Disparité à partir de l'image de disparité ce fait en trois étapes :

- **Première étape** : à partir d'un pixel de l'image de disparité, on calcule leur numéro de ligne et numéro de colonne pour lire la donnée lui correspondante dans le buffer.
- **Deuxième étape** : On incrémente la donnée du buffer puis on la réécrit dans le buffer.
- **Troisième étape** : l'enregistrement des données de buffer dans la RAM de V-Disparité.

4. Implémentation de la transformée de Hough

Nous avons vu au premier chapitre que l'extraction des segments de droite de l'image de V-disparité se fait donc par une recherche de droites significative détectée dans le tableau accumulateur. Une droite significative dans l'image de v-disparité se caractérise par un pic dans le tableau accumulateur et pour chaque ligne détecté on élimine l'effet des points appartenant à ce pic dans le tableau accumulateur. Le résultat de l'extraction est une liste de segment des droites où chaque segment est stocké avec la liste d'attributs suivants :

- Index : un entier qui caractérise sa position dans la liste des droites.
- Sa distance perpendiculaire par rapport au repère de l'image.

- Son orientation dans ce repère.

L'organigramme de la **figure 2.11** que décrit le calcul de la transformée de Hough est implémenté sur software avec les caractéristiques suivante :

- Micro ordinateur pentium 4, processeur 2.8 GHZ de vitesse, RAM 512 MB, Carte graphique de 64 MB.
- L'environnement de programmation **Microsoft Visual Studio 2008,** avec le langage **C#** comme langage de programmation**.**

Les trois seuils seuilH, seuilC et seuilD sont choisis pour avoir une bonne détection des droites significatives avec :

- seuilH = seuilC = 10.
- seuilD = 4.

5. La détection des obstacles

Une fois la transformée de Hough est calculée et les droits détectés, il est aisé de déduire géométriquement diverses informations utiles pour la détection d'obstacles :

- La ligne de contact obstacle-chaussée est localisée à l'intersection entre la droite de la route et celle de l'obstacle.
- La distance D d'un obstacle donnée par **l'équation (1.2)** est calculée à partir de l'équation de droite d'obstacle et celle de la route et le point de contact obstacle-chaussée.

Les paramètres du capteur stéréoscopique sont choisis de la manière suivante :

- La base stéréoscopique b = 1.03 m.
- La hauteur du capteur par rapport au sol H= 1.4 m.
- L'angle de tangage Φ= 9.75°.
- Focale de la caméra f=8.5 mm.
- Taille des pixels en V = Taille des pixels en U = 7.2µm.
- La résolution de chaque image dépend de la taille des images stéréoscopiques.

Cette configuration permet l'exploration d'un intervalle de disparité de [0,150] pixels.

6. Conclusion

Cette partie avait pour objectif de développer des architectures massivement parallèles capables, à partir de deux images monochromes droite et gauche, de générer des cartes de V-disparités en temps réel à la cadence de l'horloge pixel des caméras. Pour atteindre cet objectif nous avons évalué une solution en tenant compte de l'aspect « temps de développement » et performances « temps réel ». Nous avons opté pour une implémentation de l'architecture dans un circuit FPGA avec une description des architectures de traitement la plus proche possible du matériel lorsque les contraintes l'imposaient et exploité les fonctions précâblées disponibles dans le circuit. Moyennant ces différents points l'architecture globale présentée dans ce chapitre nous a permis d'atteindre notre objectif. Par ailleurs l'utilisation du langage Handel-C comme outil programmation et ses possibilités de génération nous a permis d'obtenir un certain niveau de réutilisabilité des différentes architectures ; c'est toutefois limité par l'utilisation de fonctions spécifiques précâblées (blocs mémoire essentiellement).

La transformée de Hough et la détection des informations utiles de l'obstacle sont implémentées sur micro ordinateur pentium 4 à cause du grand nombre d'opérations arithmétiques lourdes et coûteuses en ressources.

Résultats & Simulations

1. Résultats de simulation sur MATLAB

La simulation sur MATLAB se fait avec le schéma de simulation suivant :

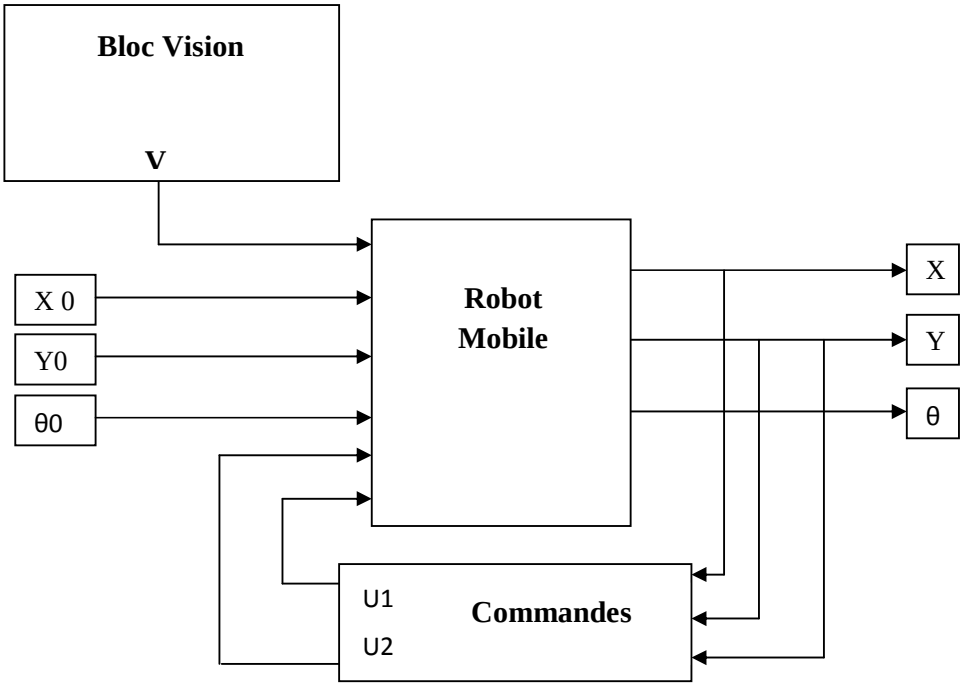

Figure 6.1 Schéma bloc de simulation sur MATLAB

Cette simulation se fait en deux étapes :

- **La première étape** sans intervention du bloc vision.
- **La deuxième étape** avec l'intervention du bloc vision.

1.1 Simulation sans intervention du bloc vision

A partir de l'étude réalisée précédemment, nous illustrons le comportement du robot (observer les états de système) en appliquant les commandes stabilisantes données par les équations (4.32) et (4.33).

$$u_1(x) = -k_1 x_1 \qquad (4.32)$$

$$u_2(x) = -k_2 x_1 - k_3 x_2 - \frac{C_3 S_3}{x_1} \qquad (4.33)$$

Nous testons les performances du contrôleur à différentes conditions initiales (x_0, y_0, θ_0)

Pour appliquer ces commandes nous fixons d'abord la valeur des différents termes de chaque commande.

Les valeurs des $k_i (1 \leq i \leq 3)$ sont données comme suit : $k_1 = 1, k_2 = 0, k_3 = 3$.

Nous avons, de ce fait, déduire les valeurs de k_a, k_b et C qui valent :

$k_a = k_1 - k_3 = -2$, $k_b = k_1 + k_3 = 4$.

$C > k_1 + k_3 \Rightarrow C > 4$ (On prend $= 10$).

Nous effectuons dans ce qui suit des tests de simulation sur le robot. L'objectif est de visualiser sa trajectoire et sa position instantanée en appliquant la commande stabilisante.

Aussi, nous intéressons à observer le comportement des commandes en rotation et en translation pendant le mouvement du robot.

- **Pour les conditions initiales ($x_0 = -5$, $y_0 = -3, \theta_0 = \pi/4$).**

  - Les allures x, y sont données par les figures suivantes :

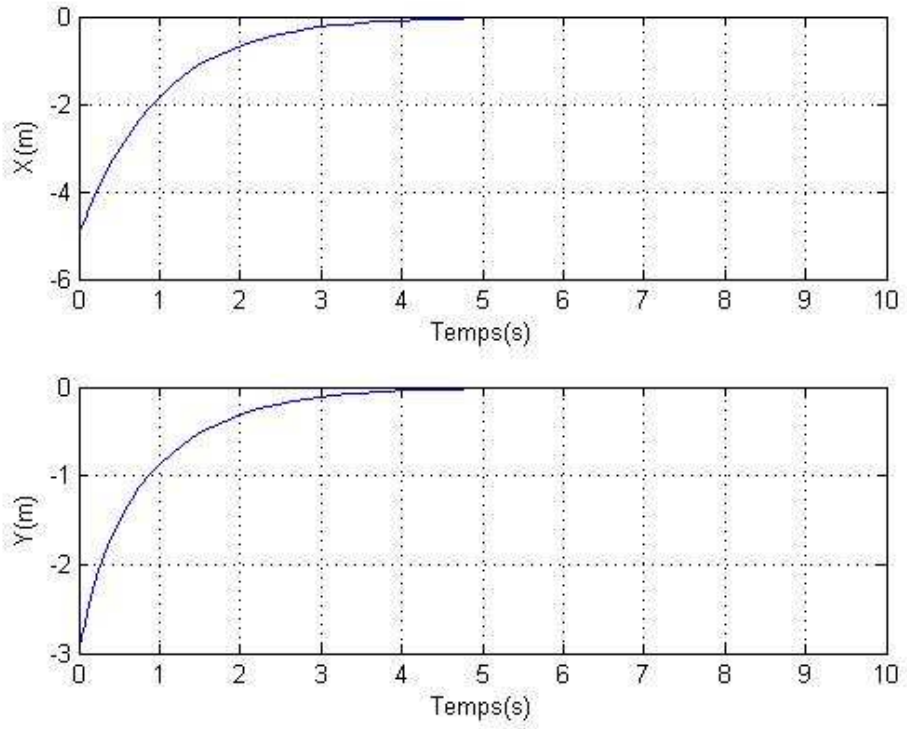

Figure 6.2 Evaluation de x(t) et y(t).

- Le comportement des commandes stabilisantes est comme indiqué dans les figures suivantes :

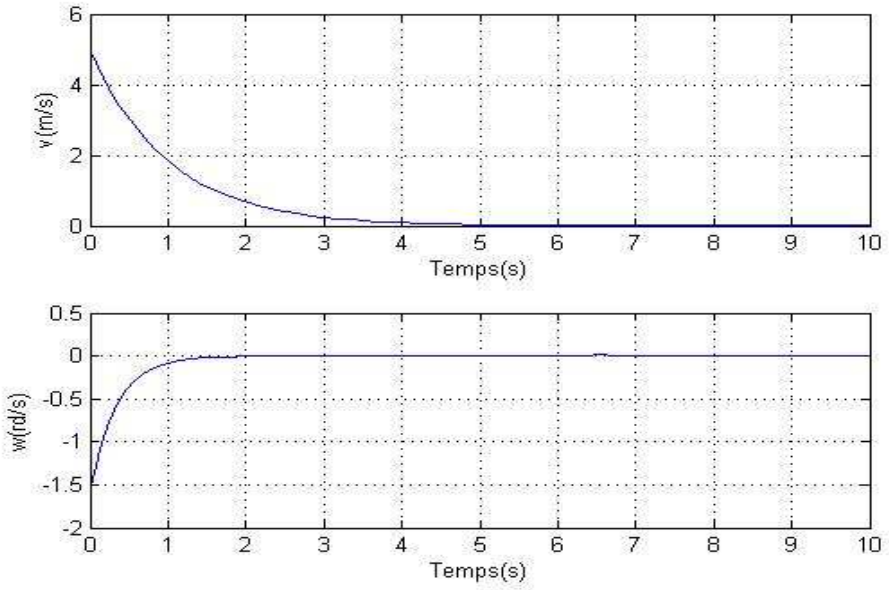

Figure 6.3 Evaluation des commandes de robot.

- La trajectoire du robot pour atteindre la position finale (x=0, y=0) à partir de position initiales (x=-5, y=-3) est donnée par la figure suivante :

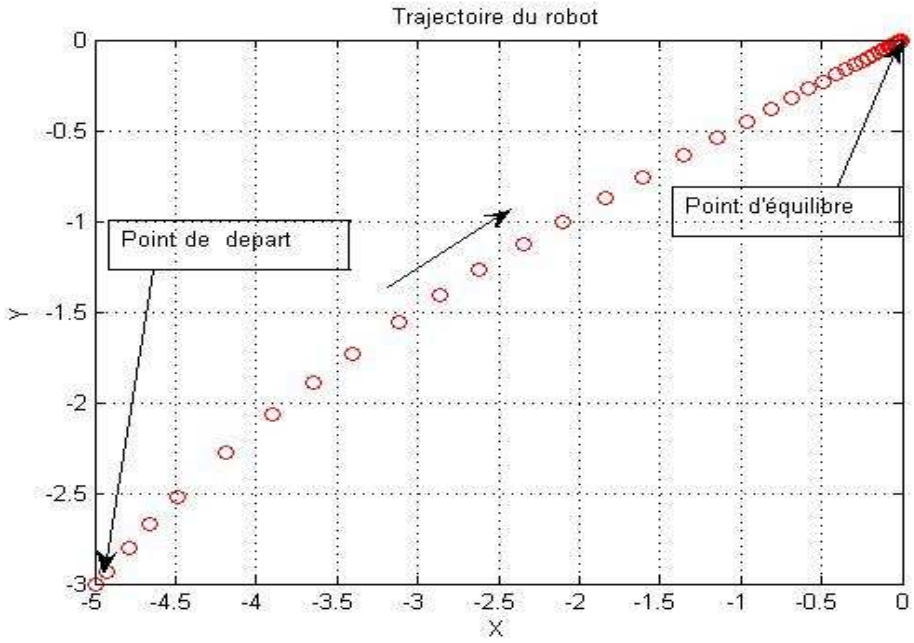

Figure 6.4 Trajectoire du robot

- **Pour les conditions initiales ($x_0 = 4$, $y_0 = 3, \theta_0 = \pi/4$).**

    - Les allures x, y sont données par les figures suivantes :

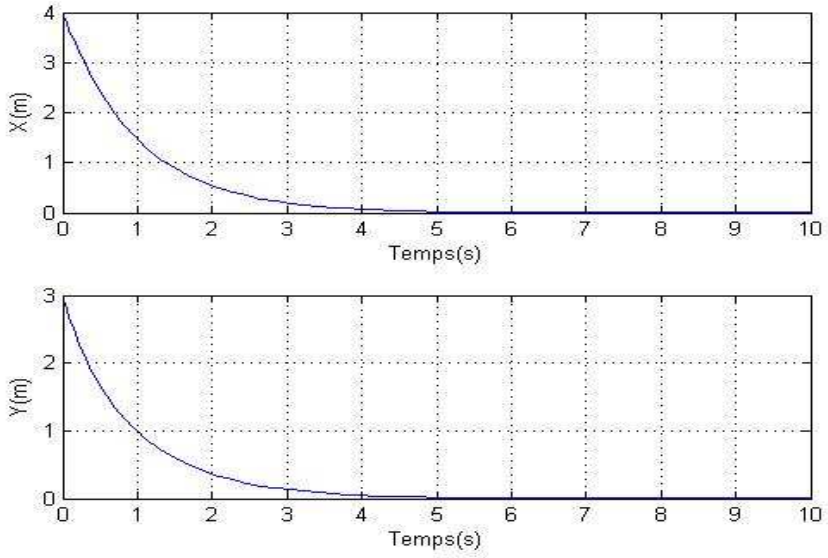

Figure 6.5 Evaluation de x(t) et y(t).

    - Le comportement des commandes stabilisantes est comme indiqué dans les figures suivantes :

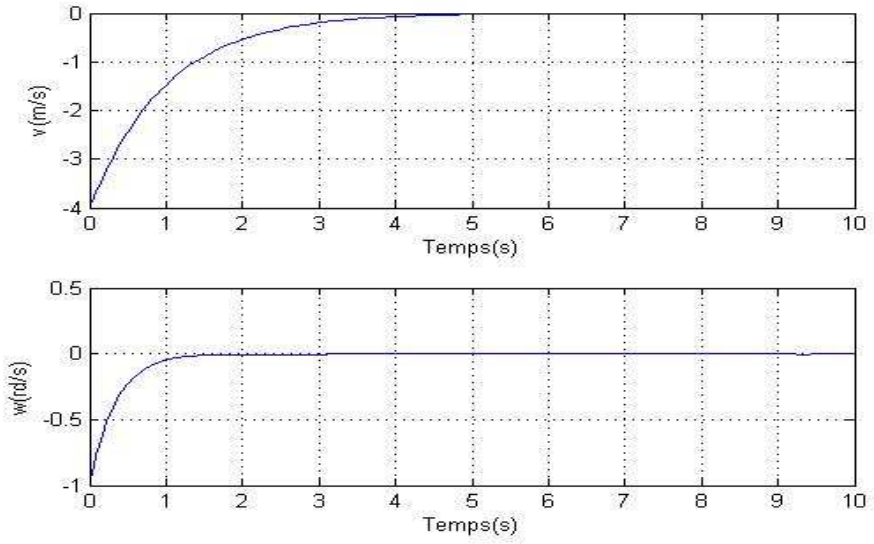

Figure 6.6 Evaluation des commandes de robot.

- La trajectoire du robot pour atteindre la position finale (x=0, y=0) à partir de position initiales (x=4, y=3) est donnée par la figure suivante :

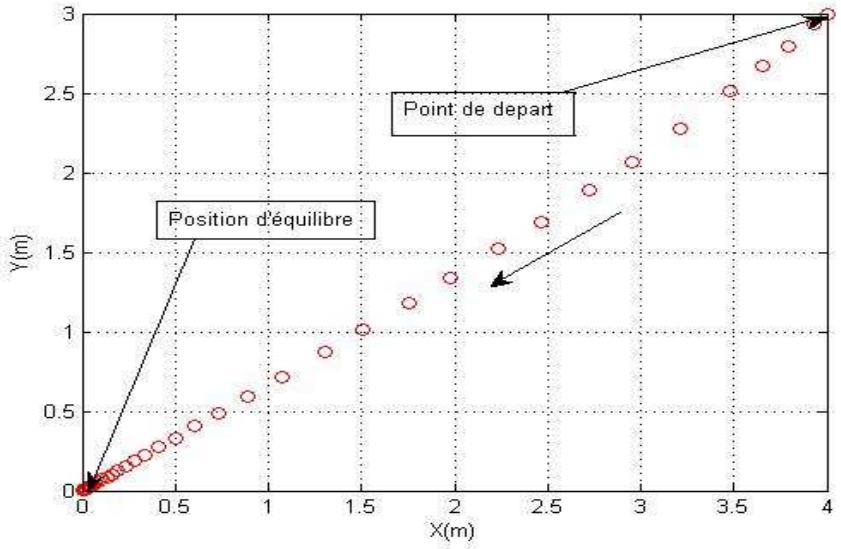

Figure 6.7 Trajectoire du robot

En résumé, nous avons effectué des tests de simulations pour observer le comportement du robot pour différentes conditions initiales.

Les résultats présentés précédemment montrent que la trajectoire de robot est converger vers l'origine à partir de n'import quelles conditions initiales.

Donc nous avons proposé une commande par retour d'état pour stabiliser un système non-holonome. Les résultats ont montré que le système converge bien vers la position d'origine et ceci quelque soit sa position initiale dans laquelle il se trouve.

1.2 Simulation avec intervention du bloc vision

Le bloc vision est un bloc que en l'utilise pour simuler notre système de vision la sortie de ce bloc est un variable binaire V (V=1 si pas d'obstacle et V= 0 s'il y a un obstacle).

On a simulé deux cas pour l'intervention du bloc vision, dans la première on arrête le robot en cas d'obstacle jusqu'à la disparition de l'obstacle et dans la deuxième on fait un évitement d'obstacle

1.2.1 L'arrête de robot en cas d'obstacle

Notre but est d'arrêté le robot en cas d'obstacle (V= 0) jusqu'à la disparition de l'obstacle, pour ce la en utilise deux nouvelles commandes U1 et U2 tel que :

$$U_i = \begin{cases} u_i & si\ V = 1 \\ 0 & si\ V = 0 \end{cases}$$

Ces deux commandes permets d'arrête le robot en cas d'obstacle et de remarche le robot s'il n y a pas d'obstacle pour attendre la position d'origine.

- **Pour les conditions initiales ($x_0 = -5$, $y_0 = -3, \theta_0 = \pi/4$) et un intervalle de détection [1,3].**

- Les allures x, y et V sont données par les figures suivantes :

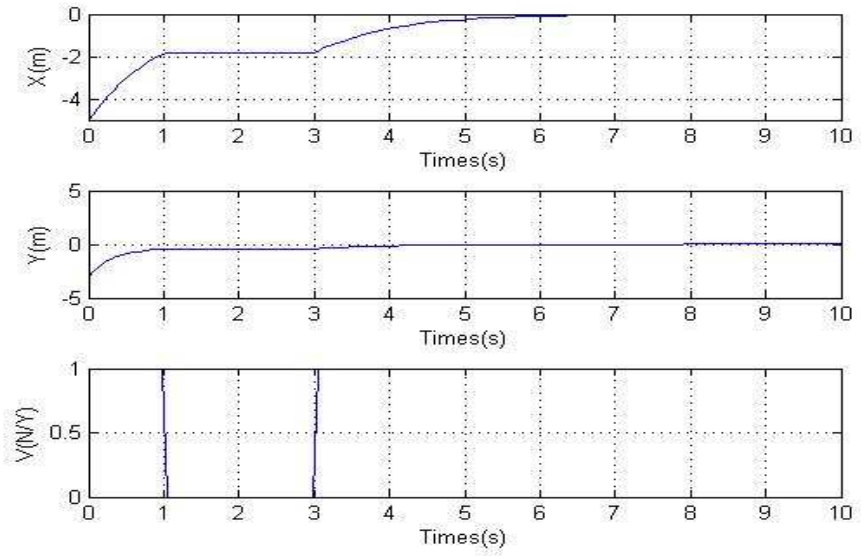

Figure 6.8 Evaluation de x(t), y(t) et V(t).

- Le comportement des commandes stabilisantes est comme indiqué dans les figures suivantes :

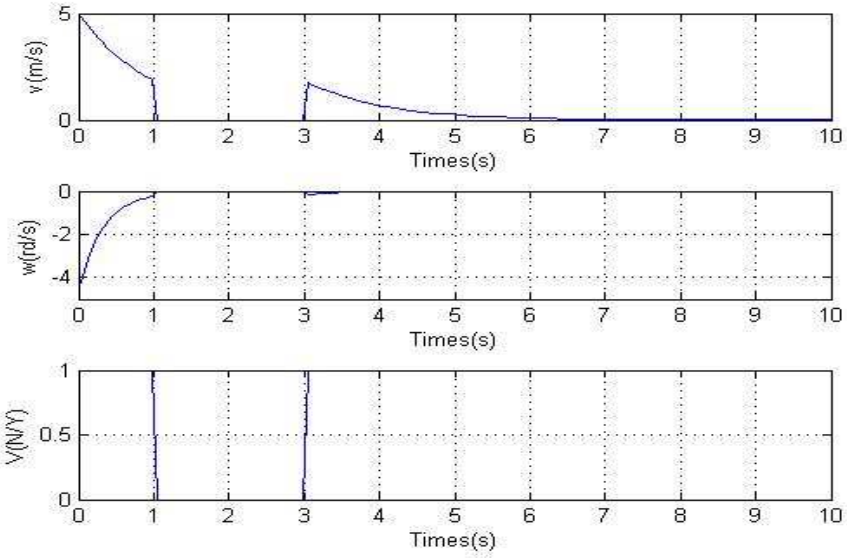

Figure 6.9 Evaluation des commandes de robot.

- La trajectoire du robot pour atteindre la position finale (x=0, y=0) à partir de position initiales (x=-5, y=-3) est donnée par la figure suivante :

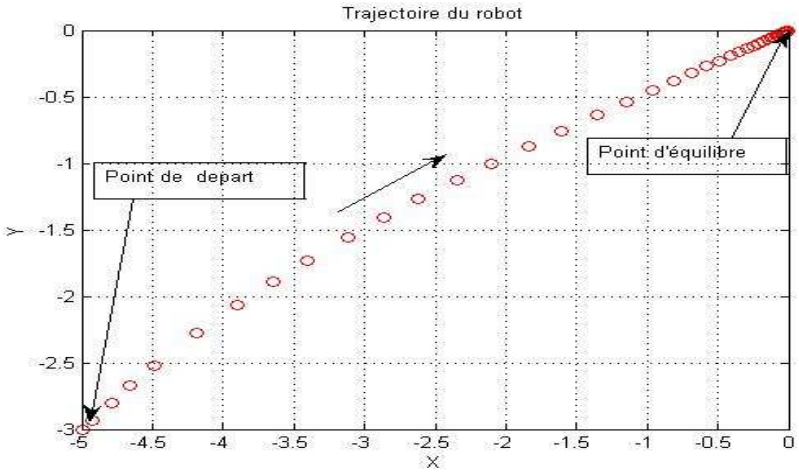

Figure 6.10 Trajectoire du robot

- **Pour les conditions initiales $(x_0 = 4, y_0 = 3, \theta_0 = \pi/4)$ et un intervalle de détection [1,3].**

- Les allures x, y et V sont données par les figures suivantes :

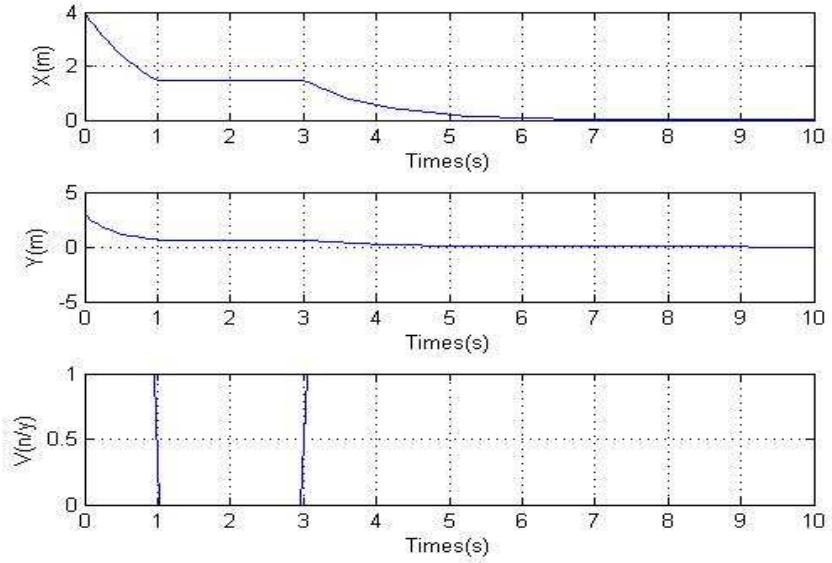

Figure 6.11 Evaluation de x(t), y(t) et V(t).

- Le comportement des commandes stabilisantes est comme indiqué dans les figures suivantes :

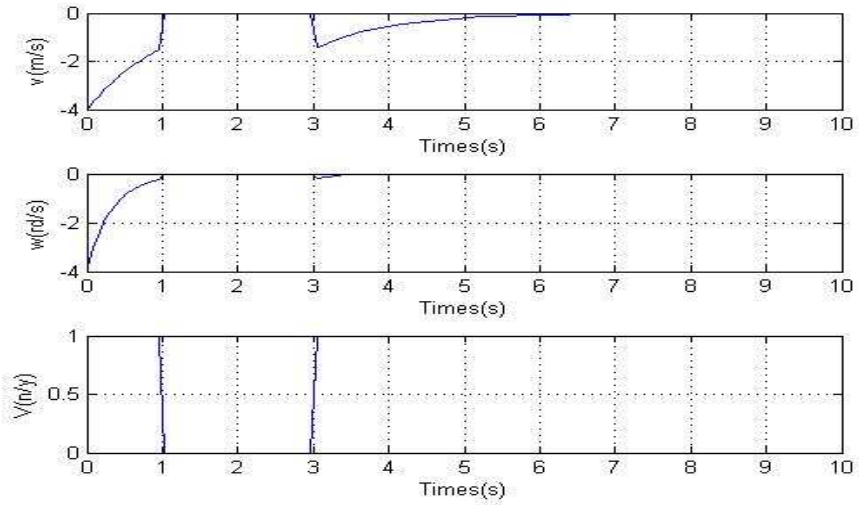

Figure 6.12 Evaluation des commandes de robot.

- La trajectoire du robot pour atteindre la position finale (x=0, y=0) à partir de position initiales (x=4, y=3) est donnée par la figure suivante :

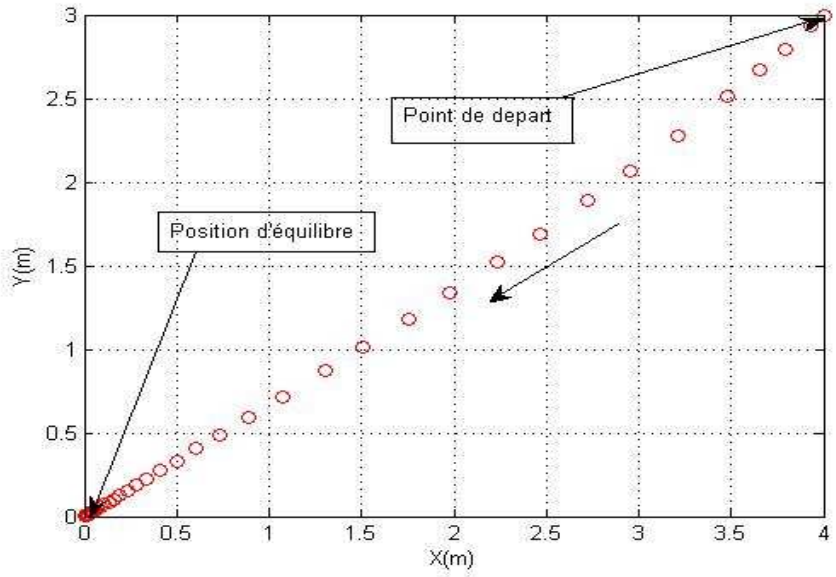

Figure 6.13 Trajectoire du robot

1.2.2 L'évitement d'obstacle par vision

Notre but est de changer la direction de robot en cas d'obstacle (V= 0) jusqu'à la disparition de l'obstacle, pour ce la en utilise deux nouvelles commandes U1 et W1 tel que :

$$U_1 = \begin{cases} U \ \ si\ V = 1 \\ Cst \ \ si\ V = 0 \end{cases}$$

$$W_1 = \begin{cases} W \ \ si\ V = 1 \\ W \pm \Delta w \ \ si\ V = 0 \end{cases}$$

Tel que Δw est un terme permet la petit déviation de robot.

Ces deux commandes permets la déviation de robot en cas d'obstacle et de rediriger le robot s'il n y a pas d'obstacle pour attendre la position d'origine.

• Pour les conditions initiales ($x_0 = -5$, $y_0 = -3, \theta_0 = \pi/4$) et un intervalle de détection [1,2].

- Les allures x, y et V sont données par les figures suivantes :

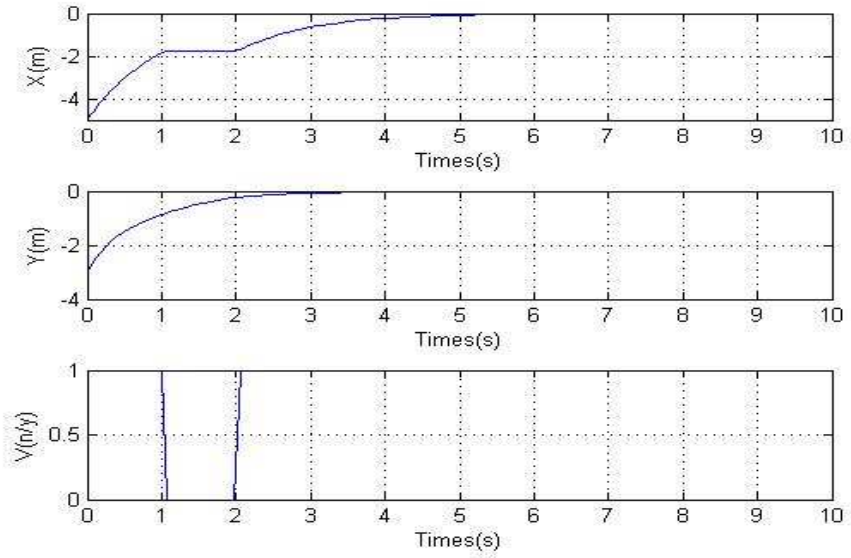

Figure 6.14 Evaluation de x(t), y(t) et V(t).

- Le comportement des commandes stabilisantes est comme indiqué dans les figures suivantes :

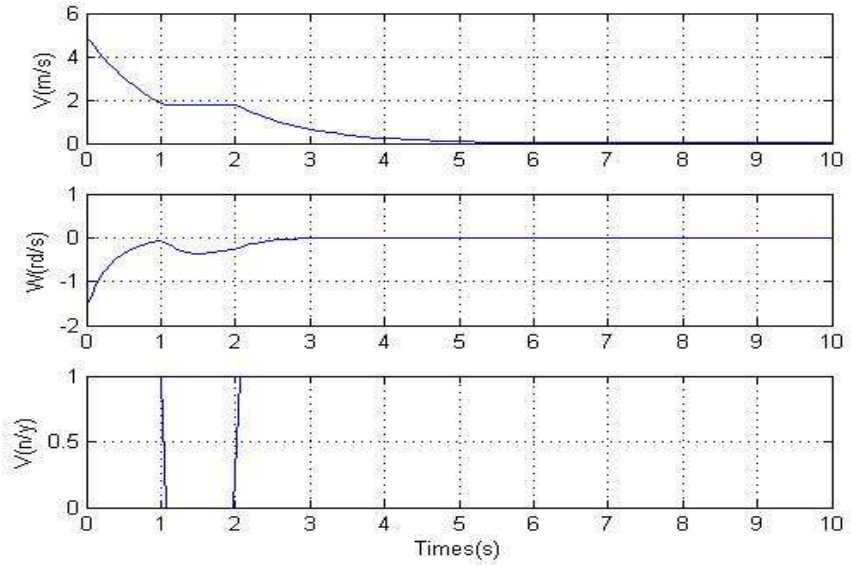

Figure 6.15 Evaluation des commandes de robot.

- La trajectoire du robot pour atteindre la position finale (x=0, y=0) à partir de position initiales (x=-5, y=-3) est donnée par la figure suivante :

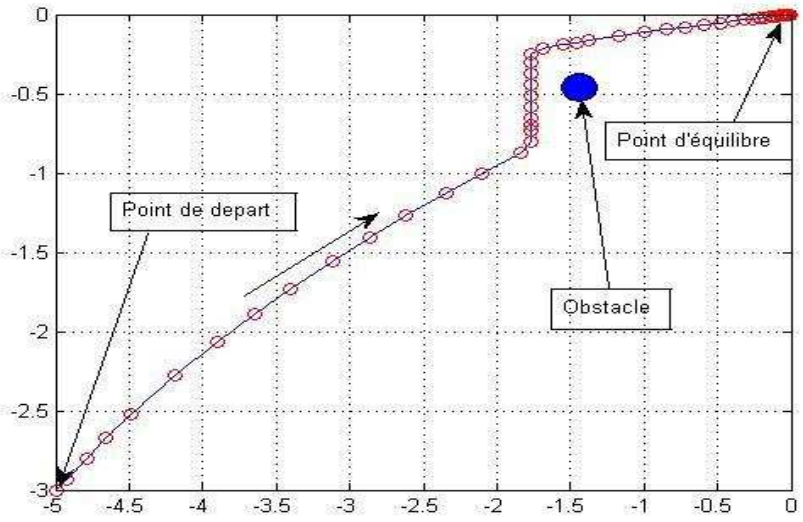

Figure 6.16 Trajectoire du robot

• Pour les conditions initiales $(x_0 = 5, y_0 = 3, \theta_0 = \pi/4)$ et un intervalle de détection [1,2].

- Les allures x, y et V sont données par les figures suivantes :

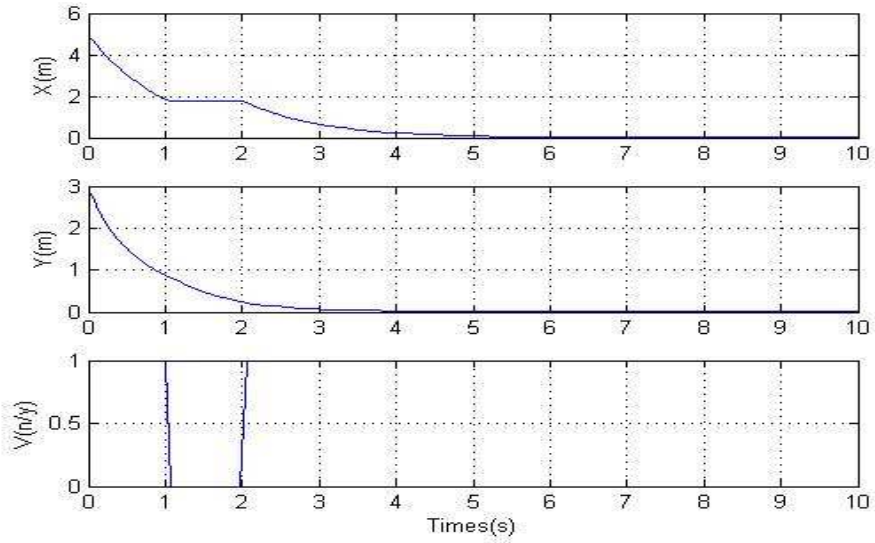

Figure 6.17 Evaluation de x(t), y(t) et V(t).

- Le comportement des commandes stabilisantes est comme indiqué dans les figures suivantes :

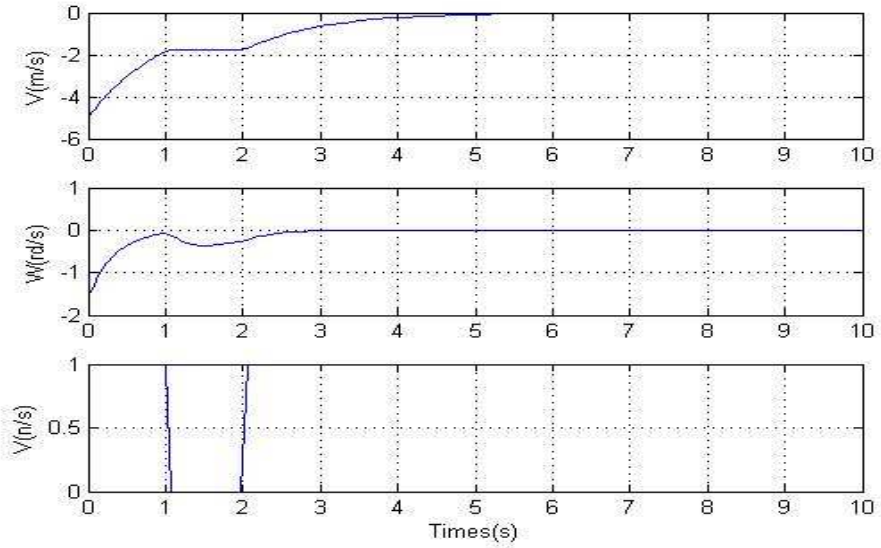

Figure 6.18 Evaluation des commandes de robot.

- La trajectoire du robot pour atteindre la position finale (x=0, y=0) à partir de position initiales (x=4, y=3) est donnée par la figure suivante :

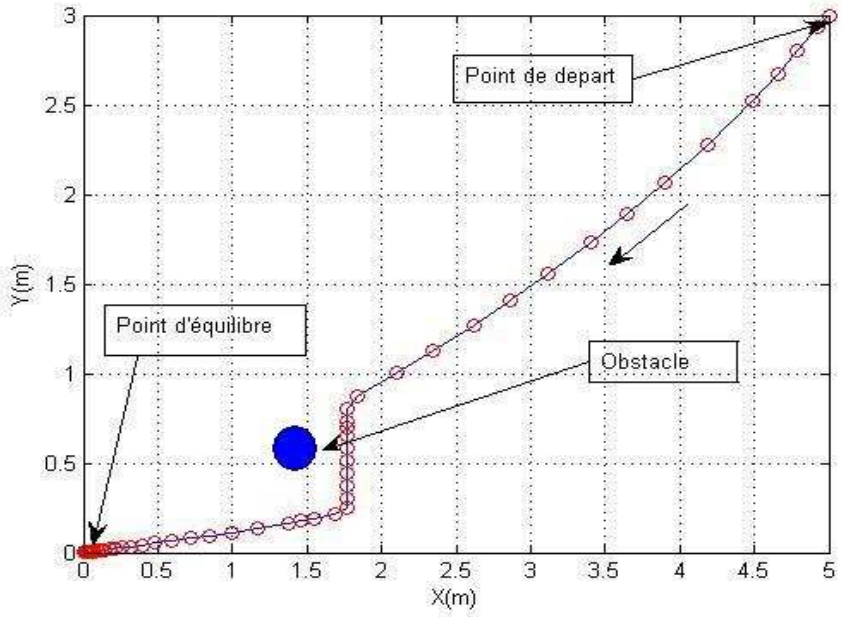

Figure 6.19 Trajectoire du robot

2. Résultats d'implémentation sur Visual Studio 2008

L'organigramme de la **figure 5.1** que décrit l'approche V-disparité est implémenté sur software avec les caractéristiques suivantes :

- Micro ordinateur pentium 4, processeur 2.8 GHZ de vitesse, RAM 512 MB, Carte graphique de 64 MB.
- L'environnement de programmation **Microsoft Visual Studio 2008,** avec le langage **C#** comme langage de programmation**.**

Les résultats d'implémentation de chaque étape sont donnés sur les figures suivantes.

2.1 La fenêtre principale de l'application

La fenêtre principale de l'application c'est la fenêtre par default de l'application, elle est comme suite :

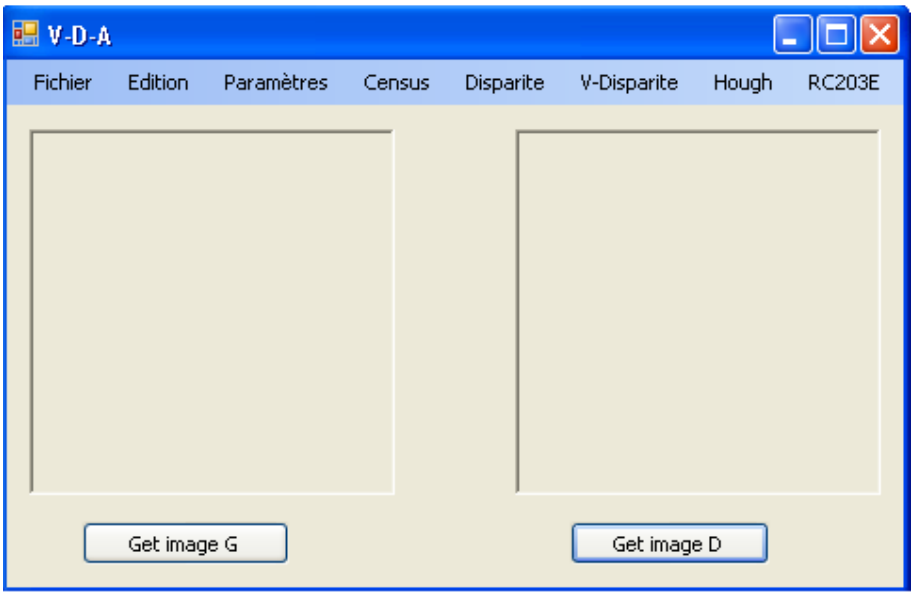

Figure 6.20 La fenêtre principale de l'application

2.2 La fenêtre Options

Dans cette fenêtre on fixe les paramètres globales pour notre procédure d'implémentation tel que : la taille des images, la taille des fenêtres Census et la disparité maximale.

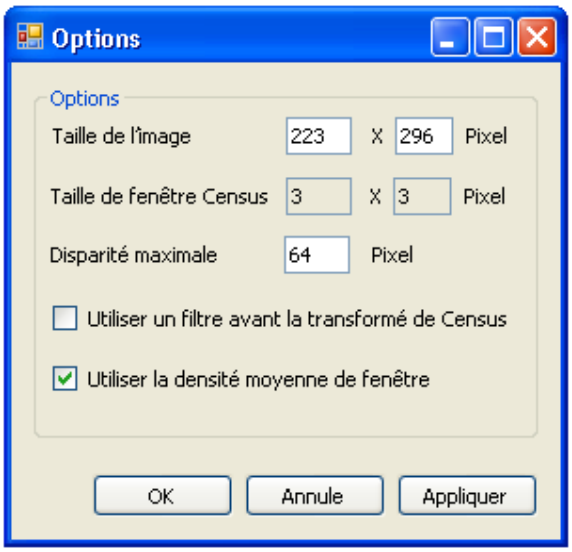

Figure 6.21 La fenêtre Options

2.3 L'acquisition des images

L'acquisition des images gauche et droite se fait par deux boutons sur la fenêtre principale de l'application.

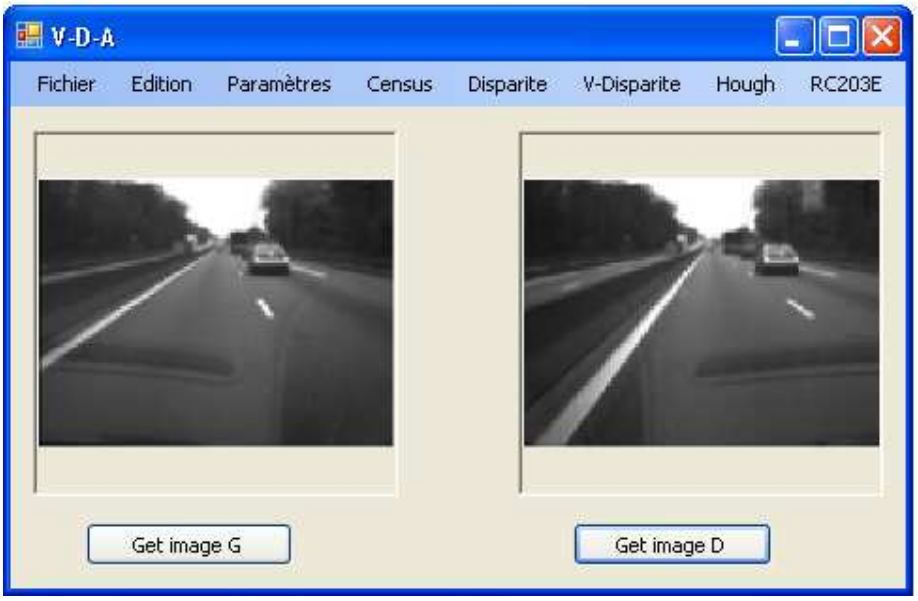

Figure 6.22 La fenêtre d'acquisition des images

2.4 La transformée de Census

Dans cette étape on calcul la transformée de Census pour les deux images gauche et droite, le résultat d'implémentation de la transformée de Census est donnée par la figure suivante :

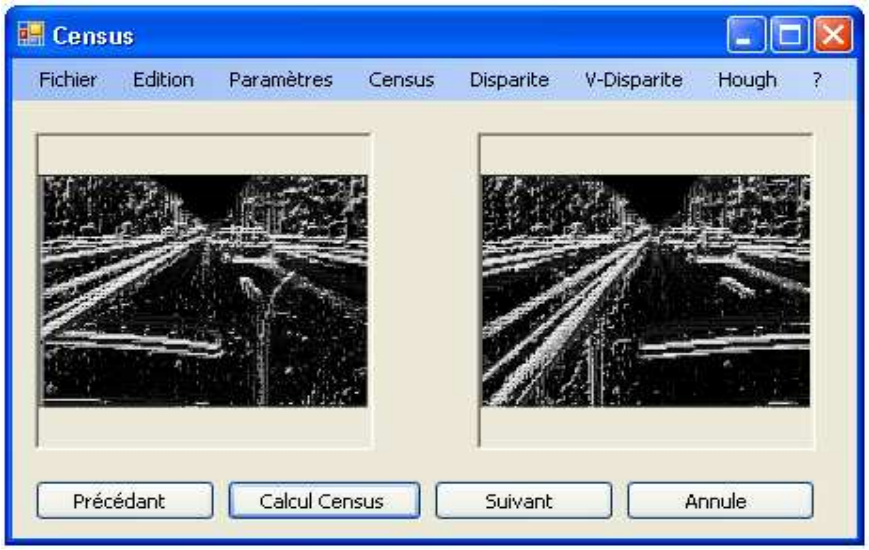

Figure 6.23 La fenêtre de la transformée de Census

2.5 Calcul de disparité

A partir des deux images Census (gauche et droite), on cherche pour chaque pixel gauche son meilleur correspondant dans une chaîne de pixels de l'image droite. Pour cela on calcule dans un premier temps pour chaque pixel candidat de l'image droite, le score associé à l'appariement : celui-ci correspond au nombre de bits identiques entre le pixel gauche et le pixel de droite concerné.

Dans un deuxième temps on effectue une recherche du meilleur score parmi les scores issus des pixels candidats.

Le résultat d'implémentation de cette procédure est donné par la figure suivante:

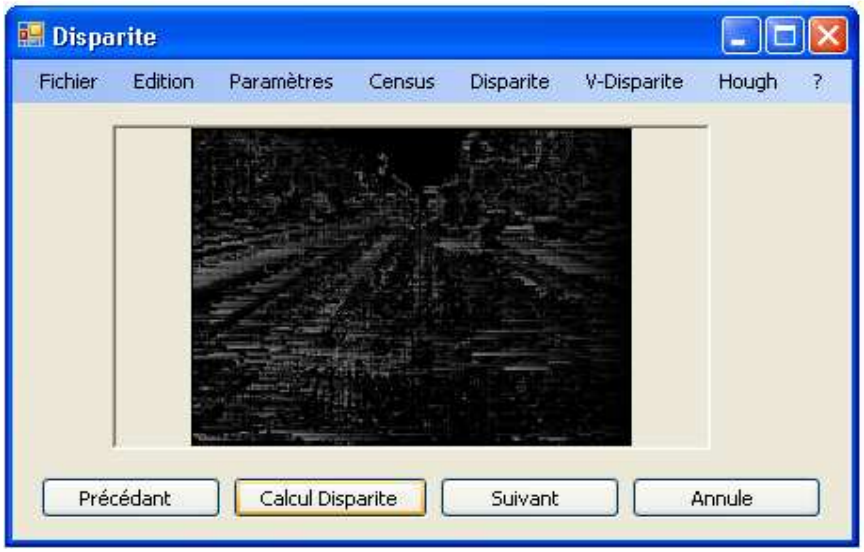

Figure 6.24 La fenêtre du calcul de disparité

2.6 Calcul de V-disparité

Une fois la carte de disparité calculée, l'image « v-disparité » est construite en accumulant pour chaque ligne image tous les pixels de disparité connue.

Ainsi, pour la ligne image i, l'abscisse u_m du point M dans l'image « v-disparité » correspond à la disparité de ce point Δ_M et la valeur de son niveau de gris correspond au nombre de points de même disparité sur la ligne i.

La figure suivante présente le résultat d'implémentation de ce processus.

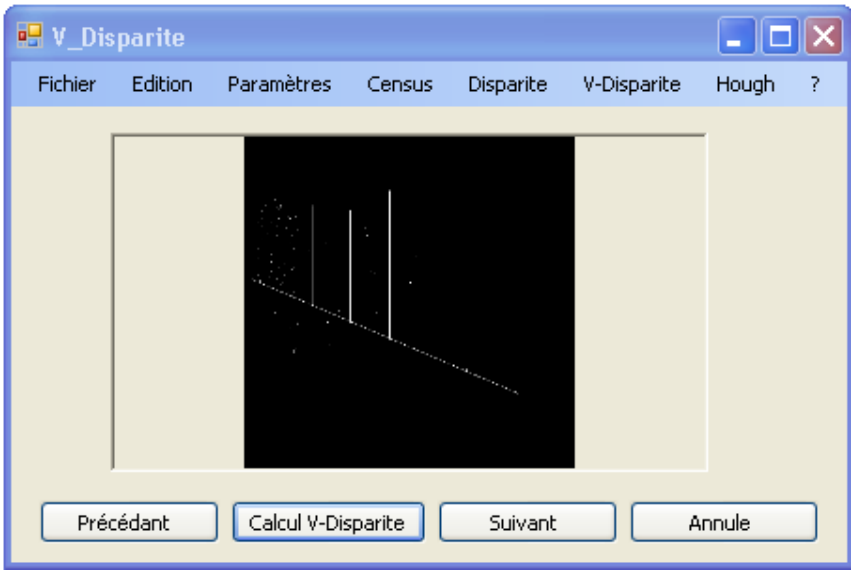

Figure 6.25 La fenêtre du calcul de V-disparité

2.7 Paramètres de modélisation du capteur stéréoscopique

La fenêtre suivante présente les paramètres de modélisation du capteur stéréoscopique. Tel que la base stéréoscopique, la hauteur du capteur par rapport au sol, l'angle de tangage entre l'axe optique et l'horizontale et le vocale de camera.

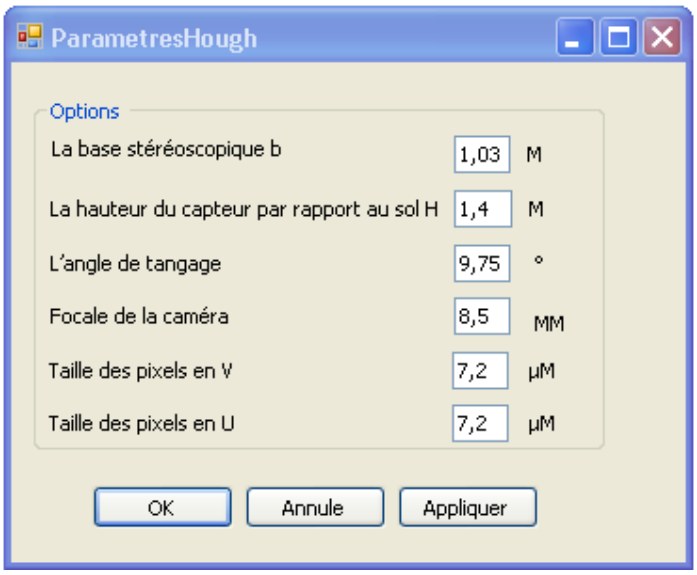

Figure 6.26 La fenêtre Paramètres du capteur stéréoscopique

2.8 La transformée de Hough et la détection des informations utiles

Dans cette étape on a implémenté l'organigramme de la **figure 2.11** que décrit le calcul de la transformée de Hough avec trois seuils : seuilH = seuilC = 10, et seuilD = 4.

Une fois les surfaces globales détectées, il est aisé de déduire géométriquement diverses informations utiles pour la détection d'obstacles tel que : la ligne de la route, les lignes verticales des obstacles et la distance de chaque obstacle.

La figure suivante présente le résultat d'implémentation de ces procédures.

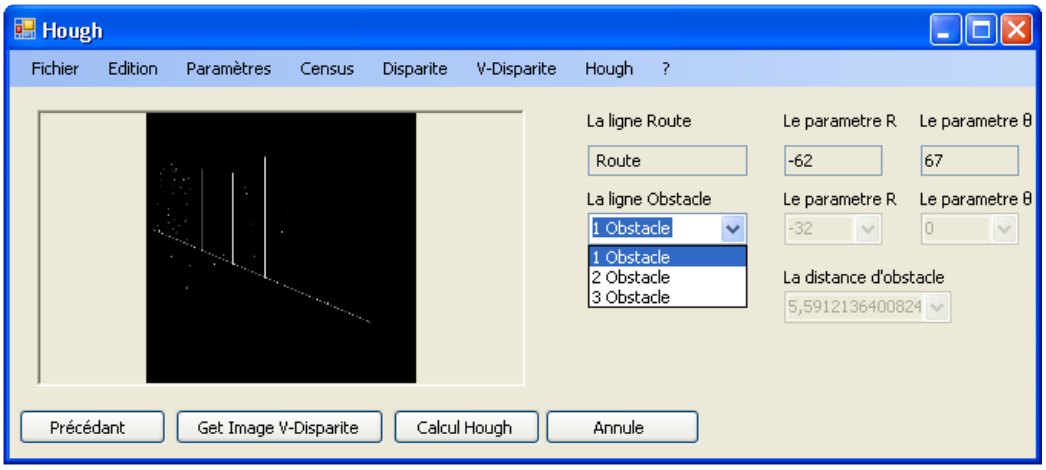

Figure 6.27 La fenêtre transformée de Hough et détection d'obstacles

3 Résultats d'implémentation de V-disparité sur FPGA

Dans cette partie de ce chapitre on présent les différents résultats d'implémentation hardware de notre architecture donnée par la **Figure 5.2** sur la carte RC203E**.**

3.1 Les ressources consommées et la fréquence maximale

Le tableau suivant représente les ressources consommées ainsi que la fréquence maximale de la détection d'obstacles par l'approche de V-disparité avec les paramètres suivantes :

- L'image de taille : 480* 640 pixels.
- La fenêtre Census de taille : 3*3 pixels.
- La disparité maximale est de : 64 pixels.

Ressources	Totale	Consommations
Entrées/Sorties(IOBs)	484	138 (28%)
LUTs	28672	3560(12%)
Slices Flips Flops	28672	1370(4%)
CLB Slices	14336	2199(15%)
RAMs		64
ROMs		16
CLKs	16	2(12%)
Fréquence MAX	-	59.24MHz

Tableau 6.1 : La consommation de ressources et la fréquence maximale pour V-disparité

❖ **Commentaire**

On remarque que la consommation de tous les types de ressources ne dépasse pas le tiers des ressources disponibles au niveau du FPGA Virtex-II, Cette implémentation utilise seulement 15% des blocs CLB disponibles, ce la peut être justifiée par le taux de parallélisme dans notre architecture surtout dans la transformée de Census.

Le taux des IOBs consommées est de 28%, cela peut être traduit par la nécessité de l'interfaçage avec les entrées sorties de la carte RC203E tel que le port parallèle, le port de camera et le port VGA.

L'architecture utilise deux horloges parmi les 16 disponible dans la carte RC203E, un pour l'acquisition des images et les autres traitements et la deuxième pour l'affichage des images sur écran VGA.

L'architecture utilise aussi 16 blocs ROMs pour l'enregistrement des configurations et 64 blocs RAMs pour les frames buffers pour le changement de la synchronisation, les lignes de retardes et le stockage des images.

L'architecture implémentée peut fonctionne en temps réel, puisque le temps du traitement (17 ns) est inférieur au tems de réponse des robots actuels.

L'outil de placement et routage trace les routes sur le circuit FPGA afin qu'il puisse réaliser le fonctionnement attendu. L'outil « FPGA Editor » nous permet de visualiser le routage réalisé par notre architecture sur la carte.

La figure 6.28 montre le routage sur FPGA du notre architecture.

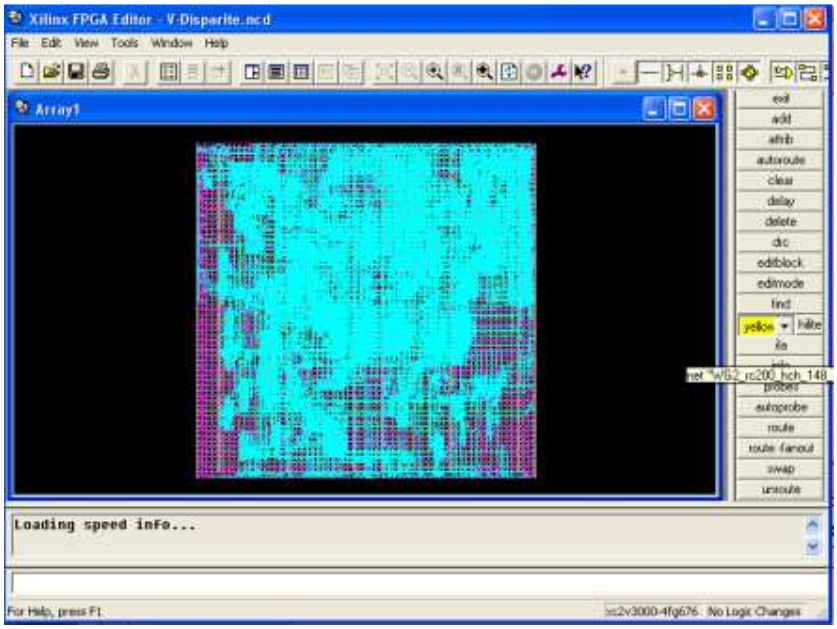

Figure 6.28 : Routage du circuit FPGA pour notre architecture

3.2 Résultats d'implémentation sur la carte RC203E

La figure suivante présent notre système de vision pour l'implémentation de notre architecture, ce système comprend :

- Une camera que ce connecte au carte RC203E par le port camera.
- Un Touchscreen intégré sur la carte que peut être utilisé comme un écran d'affichage.
- Le kit de développement RC203E.

Figure 6.29 : Aperçu de notre système de vision

L'implémentation sur la carte consiste à suivre les étapes d'implémentation décrites auparavant, les principales étapes sont les suivantes :

- **L'étape d'acquisition des images**

  L'acquisition des deux images (gauche et droite) se fait par la même camera, en premier temps l'image gauche et en deuxième temps l'image droite.

  Apres le chargement de l'image, l'image entrée sera affichée sur l'écran de la carte. La figure suivante montre l'image affichée sur la carte.

Figure 6.30 Aperçu de l'image sur l'écran de la carte

- **L'étape de la transformée de Census:**

  Apres le chargement des images sur la RAM, on calcul leurs transformées de Census. Les résultats de ce calcul seront sauvegardés sur la RAM et affichés sur l'écran. La figure suivante présente la transformée de Census de l'image précédente.

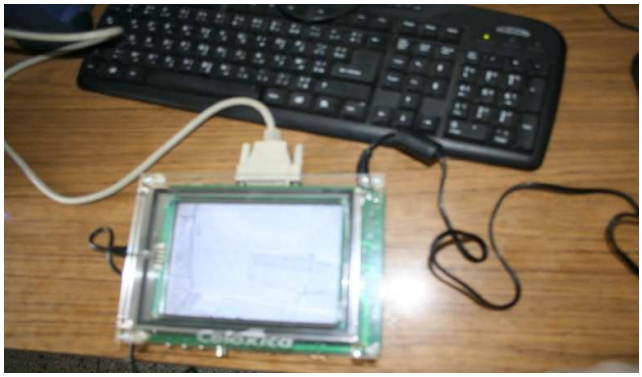

Figure 6.31 Aperçu d'une image Census sur l'écran de la carte

- **L'étape du calcul de l'image disparité**

A partir des deux images Census enregistrées sur la RAM, on calcul l'image de disparité et on l'enregistre sur la RAM. La figure suivante montre une image de disparité sur l'écran de la carte.

Figure 6.32 Aperçu d'une image de Disparité sur l'écran de la carte

- **L'étape de la génération de l'image de V-disparité**

L'image de V-disparité est calculée à partir de l'image de disparité en utilisant l'architecture proposée dans la **Figure 5.10.**

La figure suivante présente une image de V-disparité sur l'écran de la carte et avec une disparité maximale égale à 64.

Figure 6.33 Aperçu d'une image de V-disparité sur l'écran de la carte

4. Conclusion

Dans ce chapitre, la méthode de commande synthétisée dans **le chapitre 4** est testée afin de garantir la stabilisation d'un robot mobile de type unicycle.

En premier étape on test les performances de cette commande sans intervention du bloc vision, et en deuxième étape cette commande est modifiée par l'intervention du bloc vision pour garantir en première cas le freinage automatique du robot en cas d'obstacle et en deuxième cas l'évitement d'obstacle.

La simulation du bloc de détection d'obstacle ce fait en software avec **Microsoft Visual Studio 2008** et **le langage C#**.

Les résultats d'implémentation hardware du bloc de détection d'obstacles sur la carte RC203E tel que la consommation des ressources, la fréquence maximale et l'affichage des différentes étapes de détection sont données aussi dans ce chapitre.

Conclusion générale

Conclusion générale

Dans ce mémoire, nous nous somme intéressés au problème de détection d'obstacles en temps réel pour les robots mobiles en faisant appel simultanément aux techniques de détection d'obstacles par vision, les circuits logiques programmables tel que les FPGA et les techniques de commande des systèmes non linéaires tel que celles appliquées aux robots mobiles a roues.

La problématique abordée dans ce travail consiste à établir une adéquation entre un algorithme de détection d'obstacle par vision pour la navigation des robots mobiles et des architectures hardware pour l'implémentation sur un circuit FPGA. Cette adéquation est souvent limitée par les contraintes suivantes :

- **La limitation technologique** : même si un circuit FPGA représente une forte densité d'intégration, il reste tout de même limité. Il faut que les architectures proposées consomment le minimum possible des ressources ou il faut choisir correctement le composant à utiliser.

- **L'aspect temps réel :** les applications de traitement d'images sont très coûteuse en terme de calculs ce qui augmente le temps d'exécution. L'architecture proposée doit lancer plusieurs tâches en parallèle et éviter au maximum le traitement série.

- L'interprétation des algorithmes en architectures n'est pas aussi simple pour un utilisateur non expert dans la programmation des FPGAs. Ceci limite considérablement leur utilisation à grande échelle.

Afin d'atteindre nos objectifs, nous avons présenté, dans un premier, les bases théoriques de notre algorithme de détection d'obstacle tel que : la transformée de Census, le calcul de l'image de disparité, la génération de l'image V-disparité et la reconnaissance des formes par la transformée de Hough.

Nous avons ensuite établi une stratégie de commande des robots mobiles de type unicycle pour assurer leur stabilisation en un point fixe, c'est-à-dire trouver une loi de commande tel que le point d'équilibre qui représente généralement l'origine, soit stable.

Par la suite nous avons abordé notre problématique qui consiste à élaborer une implémentation hardware de notre algorithme de détection d'obstacles par vision, pour cela on a proposé une architecture puis on l'a interprétée en utilisant le langage Handel-C. On a par la suite validé notre architecture par une implémentation sur la carte RC203E dotée du composant FPGA Virtex-II. On a constaté lors de cette étape que notre architecture peut être facilement implémentée avec le respect des contraintes citées précédemment.

Pour finir ce travail, on a montré en premier temps les résultats de simulation des commandes du robot dans le cas où on n'a pas d'obstacle et dans le cas où on a un obstacle. Dans le dernier cas on a proposé deux stratégies : la première consiste à arrêter le robot en cas d'obstacle et la deuxième consiste à éviter l'obstacle par l'intervention sur les commandes internes. Par la suite on a proposé une implémentation software de notre algorithme de détection sur **Microsoft Visual Studio 2008** et **le langage C#**. Enfin on a présenté les résultats d'implémentation hardware du bloc de détection d'obstacles sur la carte RC203E tel que la consommation des ressources, la fréquence maximale et l'affichage des différentes étapes de détection.

Comme perspectives à ce travail nous proposons :

- L'augmentation de la taille de la fenêtre Census notamment lorsque les images traitées sont peu texturées. Cette opération peut entrainer des problèmes de routage dans le circuit FPGA et impliquer une reconsidération d' une partie de l'architecture.
- L'implantation d'un dispositif de filtrage d'image, lequel introduira inévitablement un retard supplémentaire dans l'apparition de l'image.
- L'implémentation d'une version incrémentale de la transformée de Hough sur la carte RC203E.
- L'utilisation de circuits FPGAs plus performants, afin de gagner en temps de calcul et de profiter de ressources fournies en plus pour implémenter toutes les étapes de détection d'obstacles et de commandes d'évitement du robot mobile.
- La validation de notre travail sur un système de vision embarqué et un robot mobile réel.

Bibliographies

[ACH04] Achour K, Djekoune O. "Incremental Hough transform: an improved algorithm for digital device implementation" . Real time Imaging, Elesevier, 2004.

[AIR94] Airiau R., Gerge JM., Olive V. « circuit synthesis with VHDL3 ». Kluwer Academic publishers, 1994.

[ASC91] Asch G. et collaborateurs, Les capteurs en instrumentation industrielle, 4ièmeédition, Dunod, 1991.

[AST96] Astolfi A, « Discontinuous control of nonholonomic system and control letters», 27:37-45, 1996.

[ATI92] Atiquzzaman M. "Multiresolution Hough transforms an efficient method of detecting pattern in images". IEEE Transactions on Pattern Analysis and Machine Intelligence; 14(11):1090-5,1992.

[BEN08] S. BENBELKACEM. " Stabilisation de systèmes non-holonomes par asservissement visuel: cas des robots mobiles à roués". Mémoires magistère, ENP ALGER 2008.

[BER09] Bernard B. "Robotique mobile". Université Louis Pasteur, 2009.

[BOU06] BOUTARFA A., « Reconnaissance de formes 3D par approche neuronale associant la transformée de Hough en robotique mobile », Thèse de doctorat, Université de Batna, 2006.

[CAM96] G. Campion, G. Bastin et B. D'Andr´ ea-Novel. Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots. IEEE Transactions on Robotics and Automation, vol. 12, no. 1, pages 47–62, 1996.

[CEL10] Disponible sur <http://www.celoxica.com>. (Consulté le 20.04.2010).

[CHA90] CHAUMETTE, F. "La relation vision-commande : théorie et application à des tâches robotiques". Thèse de doctorat, Université de Rennes 1, IRISA. Juillet 1990.

[COR96] CORKE, P. "Visual control of robots : High performance visual servoing". Mechatronics. Research Studies Press LTD. 1996. ISBN 0 86380 2079.

[CRÉ98] CRÉTUAL, A. "Asservissement visuel à partir d'information de mouvement dans l'image". Thèse de doctorat, Université de Rennes 1, IRISA. Novembre 1998.

[DAO06] H. DAOUD. Implémentation d'une commande MPPT floue sur FPGA. Mémoire de Fin d'Etudes, Electronique. Alger : Ecole Nationale Polytechnique (ENP), *2006.*

[DAV07] David F, "stratégies de commande référencées multi-capteurs et gestion de la perte du signal visuel pour la navigation d'un robot mobile", Thèse Doctorat, Université Paul Sabatier, 2007.

BIBLIOGRAPHIE

[DOL94] Dolence A. « An Overview of Rapid Prototyping Technologies in Manufacturing"Research report, Institute of Industrial Automation, Helsinki University of Techonlogy, july 1994.

[DHO89] DHOME, M., RICHETIN, M., LAPRESTE, J. et RIVES, G. "Determination of the attitude of 3d objects from a single perspective view". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no 12, pp. 1265–1278. Décembre 1989.

[DJE05] S.DJELLOULI, F. BOUNKAR. Système temps réel embarque pour commander un robot mobile. Mémoire de Fin d'Etudes, Automatique. Alger : Ecole Nationale Polytechnique (ENP), *2005*.

[DUD00] G. Dudek et M. Jenkin. Computational principles of mobile robotics. Cambridge University Press, 2000.

[DUD72] Duda R.O., Hart PE. « Use of the Hough transformation to detect lines and curves in pictures ». Communication of CAM, 15:11-15, January 1972

[FAU93] FAUGERAS, O. "Three-Dimensional Computer Vision : A Geometric Viewpoint". MIT Press, Cambridge. 1993.

[FED89] FEDDEMA, J. et MITCHELL, O. "Vision-guided servoing with feature-based trajectory generation". IEEE Transactions on Robotics and Automation, vol. 5, no 5, pp. 691–700. Octobre 1989. doi: 10.1109/70.88086.

[FOU99] J.-Y. Fourquet etM. Renaud. Coordinated Control of a Non-Holonomic Mobile Manipulator. In ISER'1999, pages 115–125, Sydney, Australie, mars 1999.

[GAU99] S. Gautama, S. Lacroix et M. Devy, 'Evaluation of stereo matching algorithms for occupant detection', International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS 99), Corfu, Grèce, 1999.

[GHO03] F.Ghozzi. « Optimisation d'une bibiothéque de modules matériels de traitement d'images. Conception et test VHDL, implémentation sous forme FPGA », These de doctorat de l'université Bordeaux I, 2003.

[HAR92] Haralick R.M., Shapiro L.G., Computer and robot vision, Vol. 1, Addison-Wesley Publishing Company, 1992

[HAR93] Haralick R.M., Shapiro L.G., Computer and robot vision, Vol. 2, Addison-Wesley Publishing Company, 1993

[HIL79] HILL, J. et PARK, W. T. "Real time control of a robot with a mobile camera". Dans 9th International Symposium on Industrial Robot. Washington, DC, pp. 233–246. Mars 1979.

[HOR89] HORAUD, R., CONIO, B., LEBOULLEUX, O. et LACOLLE, B. "An analytic solution for the perspective 4-point problem". Dans IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 47. pp. 33–44. Juin 1989.

[HOU62] Hough PVC. "Method and Means for Recognizing Complex Patterns". No.3, 1962.

BIBLIOGRAPHIE

[JUL01] C. Julien ,G. Thibault ,O. Oudry. Etude d'un algorithme de détection d'objets méthodologie de parallélisation pour FPGA. Projet de deuxième année Diplôme d'Ingénieurs. Ecole Nationale Supérieure de Physique de Straspourg . France. 2001.

[KAN86] T. Kanade, C. Thorpe et W. Whittaker. Autonomous Land Vehicle Pro- ject at CMU. In ACM Annual Computer Science Conference, pages 71–80. ACM Press, 1986.

[KHA97] KHATIB, M., JAOUNI, H., CHATILA, R. et LAUMOND, J.-P. "Dynamic path modication for car-like nonholonomic mobile robots". Dans IEEE International Conference on Robotics and Automation (ICRA'97). Albuquerque, New Mexico, USA, pp. 490–496.Avril 1997.

[KHA98] KHADRAOUI, D., DEBAIN, C., ROUVEURE, R., MARTINET, P., BONTON, P. et GALLICE, J. "Vision based control in driving assistance of agricultural vehicules". International Journal of Robotics Research, vol. 17, no10, pp. 1040–1054. 1998.

[KIT88] Kitter J., Illing J. "a survey of the Hough Transform". In computing Vision, Graphics and Image processing 44☹1),pp. 87-116,1988.

[KOB09] E.KOBZILI, « Traitmemt d'images en temps réel sur FPGA », Thèse de magistère, EMP 2009.

[KUN93] Kunt M., Granlund G., Kocher M., Traitement numérique des images, Collection Electricité, Traitement de l'Information: Vol. 2, Presses Polytechniques et Universitaires Romandes et CNET-ENST, 1999

[LAB04] R. Labayrade : «Détection générique, temps réel et robuste d'obstacles routiers par stéréovision embarquée», Thèse Doctorat, Université Pierre et Marie Curie Paris VI Jussieu, 2004.

[LAU01] LAUMOND, J.-P., rédacteur. "La robotique mobile". Hermès Sciences Publication. 2001. ISBN 2-7462-0246-86.

[LOT94] Lotufo RA., Dagless EL., Milford DJ, Morgan AD., Morrissey JF. "Hough transform for transporters arrays". In: Proceedings of the third international conference on imageprocessing and its applications, IEEE proceedings, London, pp. 122-33, 1994.

[MAL98] MALIS, E. "Contribution à la modélisation et à la commande en asservissement visuel". Thèse de doctorat, Université de Rennes 1, IRISA. Novembre 1998.

[MAR00] MARTINET, P. et THIBAUD, C. "Automatic guided vehicules : Robust controller design in image space". Autonomous Robots, vol. 8, no1, pp. 25–42. Janvier 2000. ISSN 0929-5593. doi: 10.1023/A:1008936817917.

[MAR87] Marion A., Introduction aux techniques de traitement d'images, Eyrolles, 1987.

[MAR97] Marion A. « Acquisition et Visualisation des Images ». Eyrolles, 1997.

[MEC06] C. MECHERAOUI, A. BENMOSBAH. Implémentation sur FPGA des méthodes MPPT :"P&O" et "floue optimisée par les Algorithmes Génétiques". Mémoire de Fin d'Etudes, Electronique. Alger : Ecole Nationale Polytechnique (ENP), *2006.*

[MER07] W.MERROUCHE, M. M. IDRISSA. Implémentation d'un modulateur OFDM sur un circuit FPGA. Mémoire de Fin d'Etudes, Electronique. Alger : Ecole Nationale Polytechnique (ENP), *2007*.

[MUR95] Murgai R., Brayton R., Sangiovanni-Vincentelli A. « Logic Synthesis for FieldProgrammable Gate Array ». Kluwer Academic Publishers, 1995.

[NAO06] A. NAOULOU, « Architecture pour la stéréovision passive dense temps réel : application à la stéréo-endoscopie», Thèse Doctorat, Université Paul Sabatier, France, 2006.

[NAS09] A. Nasreddine, H. Youssouf, « La détection d'obstacles par stéréovision », Mémoire de Fin d'études, école militaire polytechnique, Alger, 2009.

[NEI72] J. Neimark et N. Fufaev. Dynamics of nonholonomic systems, volume 33. Translations of Mathematical Monographs, 1972.

[NET89] Netravali A.N., Haskell B.G., Digital pictures. Representation and compression, Applications of Communications Theory Series, R.W. Lucky (Ed.), Plenum Publishing Corporation, 1989.

[NIJ90] H. Nijmeijer et A. J. Van der Shaft. Nonlinear dynamical control systems. Springer Verlag, New York, 1990.

[OFF85] Offen RJ. "VLSI Image processing". In Mc Graw Hill Book Company New York, St Louis, San Francisco, pp. 65-68, 1985.

[RAB00] C. Rabaud, « La mise en correspondance d'image stéréoscopiques», LIRMM, France.

[RAJ09] K. Rajanish, and Others. «Unleash the system on chip using FPGA and Handel-C», Shivaji University, Kolhapur India, 2009.

[RIV97] RIVES, P. et BORRELLY, J. "Visual servoing techniques applied to an underwater vehicule". Dans IEEE International Conference on Robotics and Automation (ICRA'97), vol. 3. Albuquerque, New Mexico, USA, pp. 1851–1856. Avril 1997. doi: 10. 1109/IROS.1997.656798.

[ROU69] Rosenfield A. "Picture Processing by Computer". Academic, New York, 1969.

[SAK05] S.Sakhi, « Implémentation de réseaux de neurones sur la carte à FPGA RC200 », Thèse de magistère, EMP 2005.

[SAM95] Samson C, « control of chained systems: application to path following and time-varying point-stabilization of mobile robots". IEEE Transactions on Automatic control, 40:64-77, 1995.

[SAN80] SANDERSON, A. et WEISS, L. "Image-based visual servo control using relational graph error signals". Dans IEEE International Conference on Cybernetics and Society. Cambridge, Massachusetts, pp. 1074–1077. Octobre 1980.

[SHI73] SHIRAI, Y. et INOUE, H. "Guiding a robot by visual feedback in assembling tasks". Pattern Recognition, vol. 5, pp. 99–108. 1973.

[SOM97] L. Sommelier. « Mise en correspondance d'image stéréoscopiques utilisant un modèle topologique», Thèse Doctorat, Université de Lyon I, France, 1997.

[TZV91] Tzvi Ben., Sandler M. "Analogue Implementation of the Hough Transform" . In. IEEE Proceeding-G, Vol. 138, No. 4, August 1991.

[VIT06] VITRANI, M.-A., MOREL, G., BONNET, N. et KAROUIA, M. "A robust ultrason-based visual servoing approach for automatic guidance of a surgical instrument with in vivo experiments". Dans First IEEE/RAS-EMBS Biomedical Robotics and Biomechatronics (BioRob'06). pp. 35–40. Février 2006.

[WAR83] F. W. Warner. Foundations of differentiable manifolds and lie groups. Graduate Texts in Mathematics. Springer-Verlag, 1983.

[WUN97] WUNSCH, P. et HIRZINGER, G. "Real-time visual tracking of 3D objects with dynamic handling of occlusion". Dans IEEE International Conference on Robotics and Automation (ICRA'97), vol. 4. Albuberque, Mexico, pp. 2868–2873. Avril 1997.

[ZAB94] R. Zabih et J. Woodfill, 'Non-parametric local transforms for computing visual Correspondence', 3rd European Conference on Computer Vision, pp. 151-158, Stockholm, Sweden, Mai 1994.

Annexe 1

La carte de développement RC203E

La carte RC203E est une plate-forme d'évaluation et de développement à base d'un circuit FPGA. Elle inclut le circuit Virtex-II de type **XC2V3000-4FG456C.**

La Virtex-II est une famille de circuits développés pour des applications haute performance telles que les télécommunications, l'imagerie et les applications DSP. Il possède 50 broches dont 33 peuvent être utilisées en entrées/sorties.

La carte contient également deux mémoires SRAM de 4MB. Elle présente 2 générateurs d'horloges internes. Elle contient aussi un circuit de remise à zéro « Reset » activé par un bouton poussoir.

Deux LEDs et deux afficheurs 7 segments à cathode commune sont présents sur la carte, et peuvent être utilisés durant la phase de test et de debugging. Il existe aussi 2 entrées exploitables par l'utilisateur (DIP switch) qui peuvent être mis statiquement à un état haut ou bas.

La carte possède une interface RS232, une interface JTAG pour programmer l'ISP PROM et configurer le circuit FPGA ainsi qu'un connecteur de câble parallèle qui peut aussi être utilisé pour configurer le FPGA.

Il existe également un port VGA, un port camera et un port de composition vidéo IN/OUT, une interface audio de type AC'97 compatible, un connecteur clavier PS/2, un connecteur sourie, et une mémoire flash (SmartMedia flash Memory) pour charger le fichier bit.

La carte de développement comporte aussi un circuit programmable CPLD pour la configuration/reconfiguration de la carte, un connecteur Ethernet MAC/PHY 10/100 BaseT et trois générateurs de tension (12v, 5v, 3.3v).

L'équipement de la plate forme Celoxica renferme

- La Platform Support Library (PSL).
- La Platform Abstraction Layer (PAL).
- La Data Stream Manager (DSM).
- Le FTU2 BIT utilisé pour le transfert des fichiers.

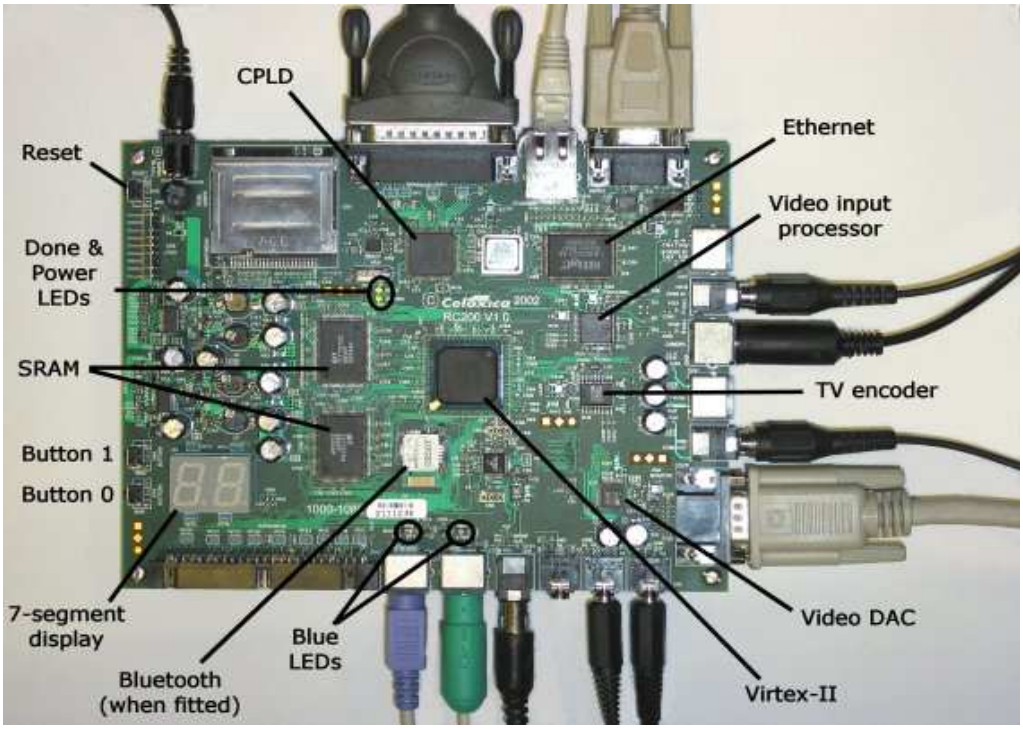

Figure 1 : Les composants de la carte RC203E

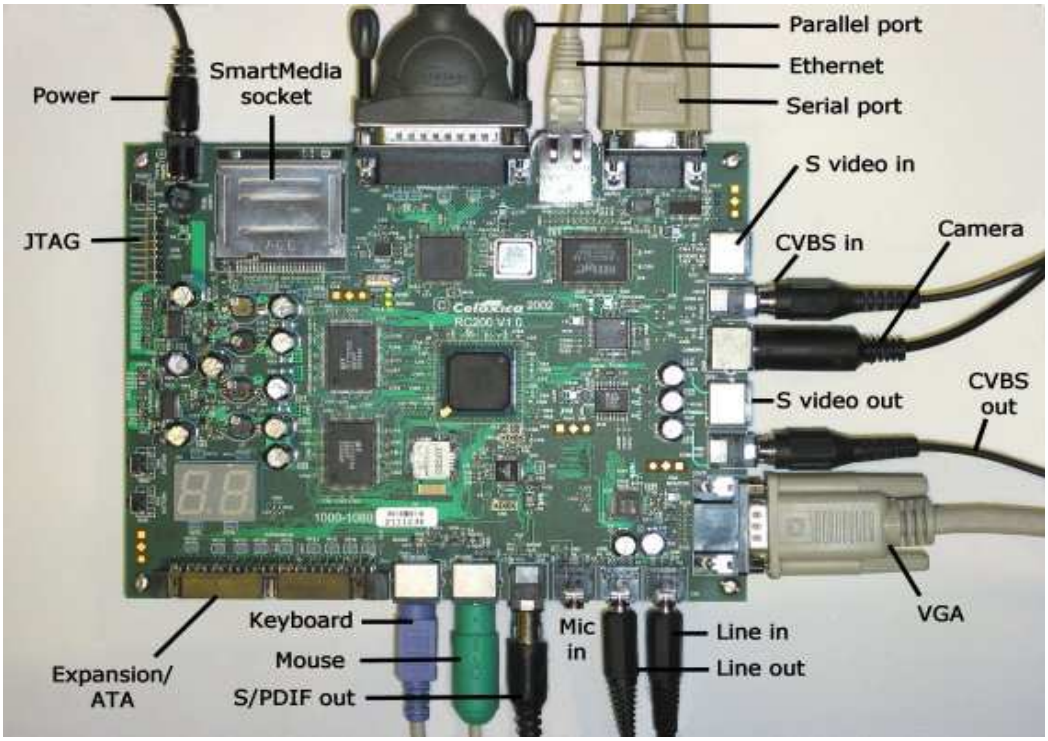

Figure 2 : Les connecteurs de la carte RC203E

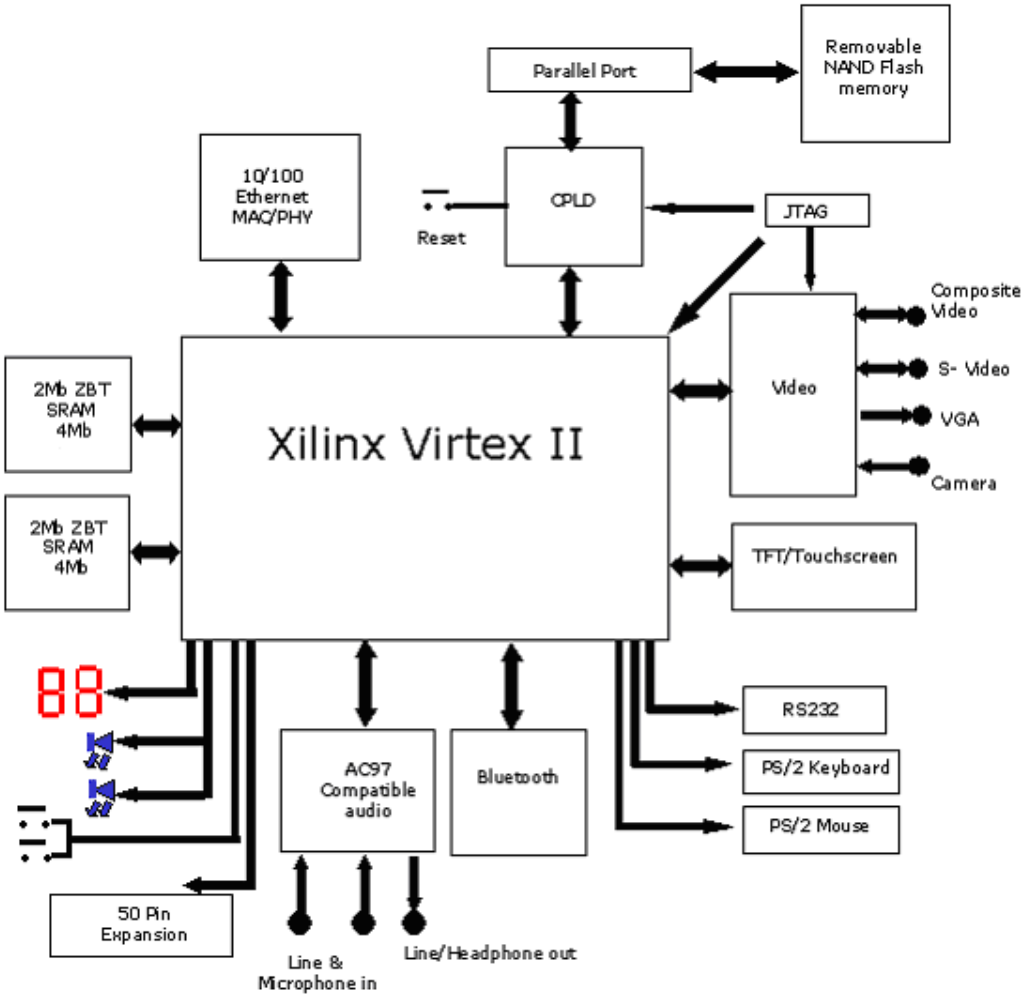

Figure 3 : Diagramme de la carte RC203E

L'outil de simulation PALSim « PAL Virtual Platform »

On a aussi une plateforme virtuelle « PAL Virtual plateforme » associée au kit pour la simulation et la visualisation des résultats, de déférentes fonctionnalités de la carte en se basant sur les APIs de PAL (La Platform Abstraction Layer). Dans le cas de traitement d'image, cette interface donne seulement le traitement effectué sur une seule image et n'est pas la vidéo. Donc elle est très utile pour la simulation fonctionnelle mais malheureusement ne montre pas l'effet de la synchronisation horizontale et verticale ce qu'elle rend que un outil de simulation et de vérification comportementale voire figure 4. Les dispositifs de la plateforme virtuel sont :

- LEDs.
- Afficheur sept segment.
- Commutateurs.
- Boutons.

- Ports génériques de données. Ceux-ci peuvent être employés pour simuler un port parallèle ou une porte série RS232. Vous pouvez choisir le fichier d'entrée et le fichier de sortie.
- Souris.
- Clavier.
- RAM rapide.
- Pipeline RAM.
- Dispositifs de sortie vidéo : vous pouvez simuler les sorties suivantes de VGA : 480 à 60 Hz régénèrent, 480 à 75 Hz régénèrent, 600 à 60 Hz régénèrent, 600 à 72 Hz régénèrent, 768 à 60 Hz régénèrent ou 768 à 76 Hz régénèrent.
- Dispositifs d'entrée Vidéo : vous pouvez simuler les entrées suivantes : 160X120, 176X144, 320X240, 352X288, 720X480 ou 720X576.
- Ethernet.
- Dispositifs d'entrée audio.
- Dispositifs de sorties audio.

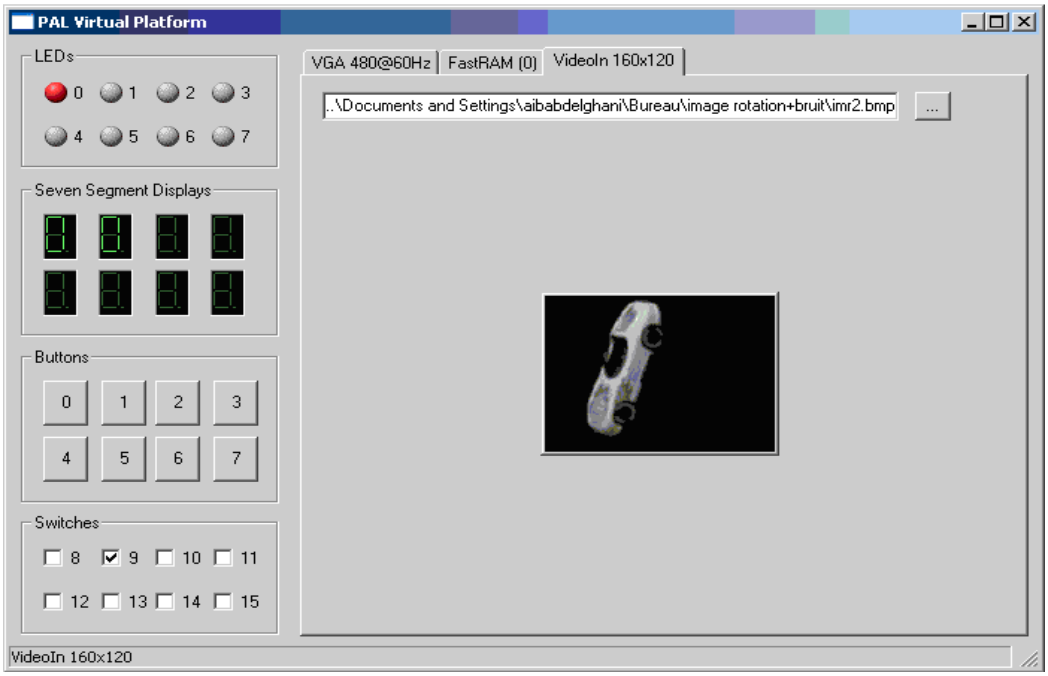

Figure 4. PAL Virtual plateforme.

Annexe 2

Les capteurs pour les robots mobiles

Nous présentons dans cette section les capteurs les plus couramment utilisés en robotique mobile pour les besoins de la navigation.

1 Les capteurs proprioceptifs

Les capteurs proprioceptifs permettent une mesure du déplacement du robot. Ce sont les capteurs que l'on peut utiliser le plus directement pour la localisation, mais ils souffrent d'une dérive au cours du temps qui ne permet pas en général de les utiliser seuls.

L'odométrie

L'odométrie permet d'estimer le déplacement à partir de la mesure de rotation des roues (ou du déplacement des pattes). La mesure de rotation est en général effectuée par un codeur optique disposé sur l'axe de la roue, ou sur le système de transmission (par exemple sur la sortie de la boite de vitesse). Le problème majeur de cette mesure est que l'estimation du déplacement fournie dépend très fortement de la qualité du contact entre la roue et le sol. Elle peut être relativement correcte pour une plate-forme à deux roues motrices sur un sol plan de qualité uniforme, mais est en général quasiment inutilisable seule pour un robot à chenille par exemple. Notons cependant que l'erreur de ces méthodes se retrouve en général principalement sur l'estimation de la direction du robot, tandis que la mesure de la distance parcourue est souvent de meilleure qualité.

Les systèmes radar doppler

Au lieu de mesurer le déplacement par des mesures sur les roues, il est possible d'utiliser un petit radar pointé vers le sol qui permet de mesurer la vitesse du véhicule par effet Doppler. Ce système présente l'avantage d'être beaucoup plus précis que la mesure passant par les roues, et d'être indépendant des dérapages possible de ces roues, mais est en général plus cher et encombrant. Il est de plus très rare sur les petites plates-formes car il ne peut mesurer de faibles vitesses de déplacement.

Les systèmes inertiels

La mesure de déplacement potentiellement la plus fiable provient de la mesure des accélérations de la plate-forme par des capteurs inertiels. Cette mesure est potentiellement fiable car elle ne dépend pas de la nature locale de l'environnement, cependant les capteurs inertiels sont tous entachés de bruit de mesure qui produit une dérive de l'estimation de la position au cours du temps.

La qualité des mesures inertielles dépend très fortement du type de capteurs utilisés. Historiquement, les premiers capteurs ont été réalisés à base de systèmes mécaniques et peuvent fournir des mesures extrêmement précises, au prix d'un coût et d'une masse très élevés. Ces dernières années ont vu apparaître de nouvelles technologies de capteurs, notamment basés sur les techniques de micro-électronique, qui ont permis la réalisation de capteurs inertiels "bas coût" et l'apparition de ces capteurs dans des produits grand public. La précision de ces capteurs est toutefois de quelques ordres de grandeur plus faible, ce qui rend leur utilisation isolée quasiment impossible. Ces capteurs fournissent toutefois un très bon complément à l'odométrie, notamment pour l'estimation de la direction.

L'accélération en translation de la plate-forme est mesurée par des accéléromètres linéaires. On dispose en général deux accéléromètres pour prendre des mesures dans deux directions perpendiculaires du plan de déplacement du robot. Un troisième peut être disposé verticalement afin de mesurer la position en trois dimensions. L'accélération angulaire est mesurée par des gyromètres. On dispose en général un gyromètre selon l'axe vertical, qui permet ainsi de mesurer l'angle de lacet du robot. Deux autres gyromètres peuvent être positionnés selon deux axes du plan de déplacement afin d'estimer la direction en trois dimensions.

Il est également possible de mesurer la rotation du robot par rapport à un axe de référence en utilisant un gyroscope. Cette mesure s'effectue en général par rapport à un axe de référence mis en rotation et isolé mécaniquement le plus possible du robot, ce qui rend sa direction indépendante de la direction du robot. Cette mesure peut être moins bruitée que l'intégration du signal d'accélération mais dépend très fortement de la qualité de la réalisation mécanique du système, qui dépend très directement du prix du gyroscope.

L'ensemble de ces éléments (3 accéléromètres et 3 gyromètres) peut être réuni pour former une centrale inertielle qui permet d'estimer complètement les six degrés de libertés de la position dans un espace à 3 dimensions. Les centrales inertielles "bas coût" sont cependant aujourd'hui de qualité insuffisante pour une utilisation isolée, tandis que les centrales de qualité correcte restent très chères. Ce domaine est cependant en évolution rapide avec l'arrivée de nouvelles technologies et l'apparition de centrales "bas coût" de qualité correcte devrait se faire dans les prochaines années.

L'utilisation des données fournies par ce type de senseurs passe aussi en général par un modèle probabiliste, qui peut être du type de celui présenté pour l'odométrie.

2 Les télémètres

Il existe différents types de télémètres, qui permettent de mesurer la distance à l'environnement, utilisant divers principes physiques.

Télémètres à ultrason

Les télémètres à ultrason sont historiquement les premiers à avoir été utilisés. Ils utilisent la mesure du temps de retour d'une onde sonore réfléchie par les obstacles pour estimer la distance. Ces télémètres sont très simple et peu cher, et sont donc très répandus, mais possèdent de nombreux inconvénients.

Télémètres à infrarouge

Ces télémètres possèdent l'avantage d'avoir un cône de détection beaucoup plus restreint. Ils utilisent une lumière infrarouge au lieu d'une onde sonore pour la détection et peuvent être basés sur différentes techniques qui permettent de recueillir plus ou moins d'information. Il est possible de mesurer simplement le retour ou le non-retour d'une impulsion codée, ce qui permet de détecter la présence ou l'absence d'un obstacle dans une certaine portion de l'espace.

Il est également possible de réaliser une triangulation sur le faisceau de retour de l'onde lumineuse, ce qui permet d'avoir une mesure de la distance de l'obstacle. Les inconvénients de ces télémètres sont liés à leur portée, en général relativement restreinte, et à leur sensibilité aux fortes sources de lumières qui contiennent un fort rayonnement infrarouge. Un projecteur du type de ceux utilisés pour la télévision pointé sur le robot, par exemple, sature en général complètement le récepteur et empêche toute détection d'obstacle. Ils sont également très sensibles à la couleur et à la nature de la surface de l'obstacle (par exemple, ils détectent difficilement les vitres).

Télémètres laser

Les télémètres les plus utilisés à l'heure actuelle pour des applications de cartographie et de localisation sont les télémètres laser à balayage. Ils utilisent un faisceau laser mis en rotation afin de balayer un plan, en général horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan (**Figure 4.7**). Cette mesure peut être réalisée selon différentes techniques (mesure du temps de retour, interférométrie...).

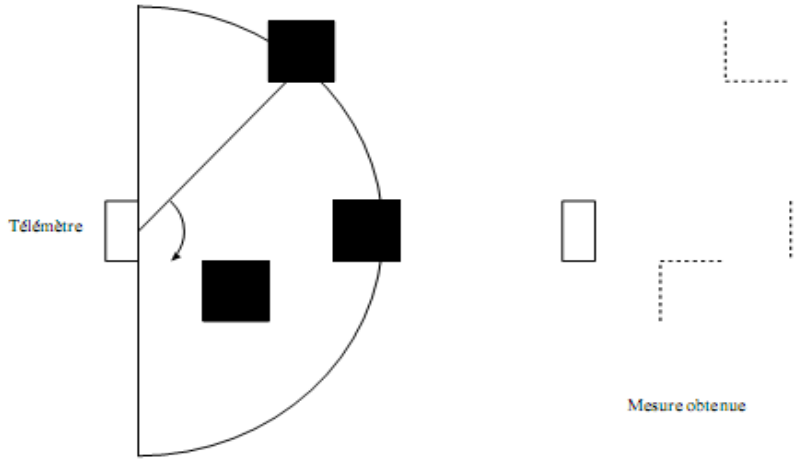

Figure. 1: Illustration d'un télémètre Laser.

Les télémètres les plus courant ont une bonne résolution angulaire car ils permettent d'obtenir une mesure de distance tout les demi degrés, sur une zone de 180 ou 360 degrés selon les modèles. La mesure est de plus relativement précise (avec un bruit de l'ordre de quelques centimètres) à une distance relativement grande (plusieurs dizaines de mètres). La fréquence d'acquisition est en général de l'ordre de la dizaine de Hertz, voire proche de la centaine pour certains modèles.

3 Les caméras

L'utilisation d'une caméra pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les humains. Le traitement des données volumineuses et complexes fournies par ces capteurs reste cependant difficile à l'heure actuelle, même si cela reste une voie de recherche très explorée.

Caméras simples

Une caméra standard peut être utilisée de différentes manières pour la navigation d'un robot mobile. Elle peut être utilisée pour détecter des amers visuels (des points particuliers qui servent de repère, tels que des portes ou des affiches) à partir desquels il sera possible de calculer la position du robot. Si ces amers sont simplement ponctuels, ou de petite taille, il sera en général simplement possible d'estimer leur direction. Dans le cas ou les amers sont des objets connus en 2 ou 3 dimensions, il sera en général possible d'estimer complètement la position du robot par rapport à la leur. Elle peut également être utilisée pour détecter des "guides" de navigation pour le robot, tels que des routes ou des couloirs.

Il est également possible d'utiliser globalement une image pour caractériser une position ou un point de vue dans l'environnement. Il faudra alors comparer cette image aux nouvelles images acquises par le robot pour savoir si le robot est revenu à cette position. Cette comparaison peut faire appel à de très nombreuses techniques, notamment à celles utilisées dans le domaine de l'indexation d'image.

Caméras stéréoscopiques

Lorsque l'on dispose de deux caméras observant la même partie de l'environnement à partir de deux points de vue différents, il est possible d'estimer la distance des objets et d'avoir ainsi une image de profondeur, qui peut être utilisée pour l'évitement d'obstacles ou la cartographie. Cette méthode suppose toutefois un minimum d'éléments saillants dans l'environnement (ou un minimum de texture) et peut être limitée, par exemple dans un environnement dont les murs sont peint de couleurs uniformes. La qualité de la reconstruction risque également de dépendre fortement des conditions de luminosité. La résolution et l'écartement des deux caméras impose également les profondeurs minimum et maximum qui peuvent être perçues, ce qui peut être limitatif pour la vitesse de déplacement du robot.

Des techniques similaires peuvent également être utilisées pour estimer la profondeur à partir d'une caméra en mouvement (méthodes de structure from motion), la difficulté étant alors d'estimer à la fois la profondeur et les positions relatives de la caméra lors de la prise des deux images.

Caméras panoramiques

Les caméras panoramiques (catadioptriques) sont constituées d'une caméra standard pointant vers un miroir de révolution (par exemple un simple cône, ou un profil plus complexe qui peut s'adapter à la résolution exacte que l'on veut obtenir sur le panorama) **(figure 4.8).** L'image recueillie permet d'avoir une vision de l'environnement sur 360 degrés autour de la camera. Le secteur angulaire vertical observé dépend de la forme du miroir et peut être adapté aux besoins de chaque application **(Figure 4.8).**

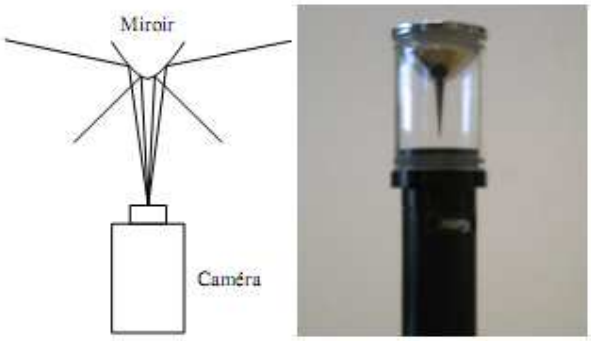

Figure 2: Principe des caméras panoramiques catadioptriques

Ce type de caméra est très pratique pour la navigation car une image prise par une camera panoramique orientée verticalement permet de caractériser une position, indépendamment de la direction du robot. En effet, pour une position donnée et pour deux orientations différentes, la même image sera formée par la caméra, à une rotation autour du centre près, tandis que pour une caméra standard, orientée horizontalement, la scène serait différente.

Ces caméras sont donc très pratiques lorsque l'on caractérise une position de manière globale, mais peuvent aussi être utilisées pour détecter des amers ou pour estimer le flux optique.

Dans ce cas, toutefois, comme la géométrie de l'image formée est relativement complexe et comme la résolution obtenue varie énormément selon la direction observée, les algorithmes doivent être adaptés, ce qui pose un certain nombre de problèmes.

Concernant le flux optique, cependant, les caméras panoramiques possèdent l'avantage de contenir toujours le point d'expansion et le point de contraction dans l'image, ce qui rend l'estimation du mouvement beaucoup plus aisée.

4 Autres capteurs

Les capteurs tactiles

Les robots peuvent être équipés de capteurs tactiles, qui sont le plus souvent utilisés pour des arrêts d'urgence lorsqu'il rencontre un obstacle qui n'avait pas été détecté par le reste du système de perception.

Ces capteurs peuvent être de simples contacteurs répartis sur le pourtour du robot. Ils ne détectent alors le contact qu'au dernier moment. Il est également possible d'utiliser des petites tiges arquées autour du robot pour servir d'intermédiaire à ces contacteurs, ce qui permet une détection un peu plus précoce et donne ainsi plus de marge pour arrêter le robot.

Les boussoles

Les boussoles permettent, par la mesure du champ magnétique terrestre, de déduire la direction du nord. Ces capteurs peuvent utiliser différentes technologies et ont l'avantage de fournir une direction de référence stable au cours du temps (au contraire des gyroscopes qui dérivent).

Ces capteurs sont toutefois très délicats à utiliser en intérieur car ils sont très sensibles aux masses métalliques présentes dans la structure des bâtiments. En pratique, on les utilise donc principalement en extérieur en apportant le plus grand soin à leur positionnement sur le robot pour éviter les influences des composants du robot, notamment les moteurs électriques.

Les balises

Dans certaines applications, il est également possible d'utiliser des balises dont on connaît la position, et qui pourront être facilement détectées par le robot, afin de faciliter sa localisation.

Des techniques très diverses peuvent être utilisées pour ces balises. On peut par exemple utiliser un signal radio, émis de manière omnidirectionnel par la balise.

Le robot sera alors équipé d'une antenne directionnelle qui lui permettra de détecter la direction des différentes balises, afin de déduire sa position par triangulation.

On peut également utiliser des codes couleurs ou des codes barres qui pourront être détectés par une caméra.

Le GPS

Les besoins de localisation étant omniprésents dans de très nombreux secteurs de la vie actuelle, l'idée d'avoir un système de localisation le plus universel possible à donné lieu à l'apparition du Global Positionning System. C'est un système de balises dont on a placé les balises sur des satellites en orbite terrestre et qui est par conséquent accessible de quasiment partout à la surface du globe. Ce système permet donc d'avoir une mesure de sa position dans un repère global couvrant la terre avec une précision variant de quelques dizaines de mètres à quelques centimètres suivant les équipements.

Ce système est cependant loin de résoudre tous les problèmes de localisation des robots mobiles. Il fonctionne en effet difficilement dans des environnements urbains, et n'est pas utilisable à l'intérieur des bâtiments. Sa précision est de plus souvent trop faible pour qu'un robot terrestre puissent utiliser ces informations seules. En pratique, il est souvent couplé à un système inertiel qui permet de palier aux pertes du signal GPS et il ne remplace de toute façon pas les capteurs du robot qui lui permettent de percevoir son environnement immédiat, qui constitue la source d'information principale pour la navigation à court terme (par exemple l'évitement d'obstacles, par opposition à la navigation à long terme qui consiste à rejoindre un but distant).