ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE EN TECHNOLOGIE DES SYSTÈMES
M. ING.

PAR
JEAN-CHRISTOPHE DEMERS

CONCEPTION ET DÉVELOPPEMENT D'UN SYSTÈME DE PERCEPTION DE L'ENVIRONNEMENT ET DE PLANIFICATION DE TRAJECTOIRE POUR UN ROBOT MARCHEUR

MONTRÉAL, LE 25 AOÛT 2002

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

- Dr. Mohamed Cheriet, directeur du mémoire et professeur
  Département de Génie de la production automatisée à
  l'École de technologie supérieure
- Dr. Thiagas Sankar, codirecteur du mémoire et professeur
  Département de Génie de la production automatisée à
  l'École de technologie supérieure
- Dr. Richard Lepage, codirecteur du mémoire et professeur
  Département de Génie de la production automatisée à
  l'École de technologie supérieure
- Dr. Jacques-André Landry, président du jury et professeur
  Département de Génie de la production automatisée à
  l'École de technologie supérieure
- M. Alain Croteau, chercheur
  Institut de recherche électrique du Québec

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 21 JUIN 2002
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

CONCEPTION ET DÉVELOPPEMENT D'UN SYSTÈME DE PERCEPTION DE L'ENVIRONNEMENT ET DE PLANIFICATION DE TRAJECTOIRE POUR UN ROBOT MARCHEUR

Jean-Christophe Demers

SOMMAIRE RÉDUIT

Ce mémoire aborde trois des principaux aspects de la problématique associée à la robotique mobile : la perception de l'environnement, la représentation virtuelle de l'environnement et la planification de trajectoire. Plus spécifiquement, ces trois aspects sont considérés pour un robot marcheur. Puisque les capacités de déplacement supplémentaires de ce robot sont directement associées au relief du sol, les éléments présentés dans ce mémoire considèrent uniquement la topographie du terrain se trouvant devant le robot.

Pour la première partie, on présente de façon détaillée la conception et le développement d'un capteur basé sur un système de vision stéréoscopique binoculaire actif. La deuxième partie présente une méthode de représentation virtuelle de l'environnement de type surface d'élévation. Enfin, la troisième partie présente l'implantation d'un algorithme de planification de trajectoire basé sur la méthode conventionnelle suivant les contours d'obstacles polygonaux. Finalement, on présente plusieurs résultats de simulation permettant d'évaluer la performance globale des systèmes développés.

CONCEPTION ET DÉVELOPPEMENT D'UN SYSTÈME DE PERCEPTION DE L'ENVIRONNEMENT ET DE PLANIFICATION DE TRAJECTOIRE POUR UN ROBOT MARCHEUR

Jean-Christophe Demers

SOMMAIRE

Ce mémoire aborde trois des principaux aspects de la problématique associée à la robotique mobile : la perception de l'environnement, la représentation virtuelle de l'environnement et la planification de trajectoire. Plus spécifiquement, ces trois aspects sont considérés pour un robot marcheur. Puisque les capacités de déplacement supplémentaires de ce robot sont directement associées au relief du sol, les éléments présentés dans ce mémoire considèrent uniquement la topographie du terrain se trouvant devant le robot.

La première partie du mémoire aborde la problématique associée à la perception de l'environnement. Une brève revue de littérature visant à identifier les capteurs susceptibles de modéliser la surface du terrain permet de cerner les besoins de conception et de développement d'un nouveau type de capteur. On présente ensuite le capteur suggéré qui est un système de vision stéréoscopique binoculaire avec lumière structurée. On montre que ce capteur, muni d'un projecteur laser projetant une matrice de points laser au sol, crée des marques de référence servant à rendre le processus de vision indépendant de la complexité de la scène et des conditions d'éclairage. Cette partie couvre de façon détaillées toutes les étapes de conception de ce capteur. Les résultats préliminaires obtenus à l'aide du prototype montrent des performances plus que satisfaisantes avec une précision de l'ordre du centimètre.

La deuxième partie du mémoire présente la problématique associée à la représentation virtuelle de l'environnement. Encore une fois, une brève revue de littérature présente les modèles de représentation les plus couramment utilisés. On propose une modélisation de l'environnement basée sur une description de type surface d'élévation. L'environnement modélisé est réduit à sa plus simple expression et indique seulement les obstacles infranchissables. On montre que la notion d'obstacle dans le cadre d'un robot marcheur prend un sens particulier et se résume aux pentes ainsi qu'aux discontinuités verticales et horizontales.

La troisième partie du mémoire expose la problématique associée à la planification de trajectoire. Après avoir succinctement passé en revue les méthodes les plus utilisées, on présente de façon détaillée l'implantation d'un algorithme. Cet algorithme est basé sur la méthode conventionnelle de planification de trajectoire suivant les contours d'obstacles polygonaux.

Finalement, on présente plusieurs résultats de simulation qui permettent d'évaluer la performance globale des systèmes développés.

REMERCIEMENTS

La réalisation d'un travail de cette ampleur ne peut être menée à terme sans le support constant de tous les gens impliqués de près ou de loin dans le projet. Cette implication peut prendre deux formes, celle des collègues qui partagent la passion du défi relevé et celle de la famille qui s'exprime par le support moral essentiel au succès d'une telle entreprise. Ainsi l'achèvement d'un tel travail ne dépend pas seulement de l'auteur, mais aussi de toutes ces personnes chères. Je profite de cette opportunité pour vous remercier tous individuellement.

J'aimerais exprimer ma reconnaissance et ma gratitude à mon directeur de maîtrise Dr. Mohamed Cheriet et à mes codirecteurs de maîtrise Dr. Thiagas Sankar et Dr. Richard Lepage. Je tiens à remercier spécialement Dr. Cheriet pour son aide précieuse et son support inconditionnel pour tous les aspects du projet. Par la même occasion, je tiens aussi à souligner le support financier du CRSNG provenant des fonds de recherche de mes directeurs et codirecteurs, plus spécialement du Dr. Sankar ainsi que l'octroi de bourses d'excellence provenant de l'École de technologie supérieure.

Mes remerciements vont aussi aux membres du jury, Dr. Jacques-André Landry et M. Alain Croteau qui ont accepté avec enthousiasme la tâche d'évaluer la qualité de ce mémoire.

Je tiens maintenant à remercier la personne sans qui ce projet n'aurait jamais pu voir le jour, mon grand ami et collègue M. Patrick Lessard. Père du projet Capra, il a su, grâce à ses multiples talents, réunir toutes les ressources nécessaires au démarrage de ce projet ambitieux. De plus, c'est grâce à sa persévérance et à ses encouragements incessants que le projet a pu atteindre un tel niveau de qualité.

Je tiens aussi à remercier tous mes amis et collègues du projet Capra. D'abord mes remerciements vont à M. Nicolas Morency qui a permis de développer et de tester plusieurs algorithmes et approches différentes. J'aimerais aussi remercier M. Guillaume Lambert, M. Alexandre Doin, M. Benoit Lagarde et finalement tous les autres membres de l'équipe qu'il serait trop long de nommer ici.

Les commanditaires de l'équipe Capra sont nombreux et ont contribué de façon significative à l'avancement considérable du projet. J'aimerais remercier spécialement :

- Matrox pour le don de l'unité de calcul 4Sight-II et la bibliothèque de programmation MIL.
- Lasiris pour le don de tout l'équipement optique qui comprend plusieurs lasers, réseaux de diffraction et filtres interférentiels.
- Point Grey Research pour la réduction considérable du prix d'achat des caméras FireFly.

J'aimerais aussi profiter de cette occasion pour remercier mon grand ami et collègue M. Sylvain DeBlois, sans qui le succès de toutes mes études universitaires n'aurait pu atteindre le même niveau d'excellence.

Maintenant mes remerciements vont aux membres de ma famille. À mes filles adorées, Karolanne et Kassandra, grands trésors de ma vie, sources d'inspiration et de bonheur intarissables, je vous remercie d'avoir accepté de façon aussi stoïque mes absences souvent prolongées. À ma chère Meybel, je tiens à te remercier pour ta patience et ta compréhension tout au long de ces années d'étude. Finalement, à mes parents et mon frère, Jean-Pierre, Louise et Jean-Sébastien, je tiens à vous remercier pour tout ce que vous m'avez apporté de merveilleux. Je tiens à remercier spécialement ma mère pour les innombrables suggestions relatives à la qualité du texte.

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS ET DES SIGLES

2D	Deux dimensions ou bidimensionnel
2D½	Représente les environnements non structurés où le chemin le plus court sur une surface n'est pas la ligne droite
3D	Trois dimensions ou tridimensionnel
4Sight-II	Ordinateur industriel conçu et fabriqué par la compagnie Matrox
Bitmap	Format numérique d'une image
CCD	*Charged Coupled Device* – Capteur d'image électronique
CDRH	*Center for Devices and Radiological Health*
CFR	*Code of Federal Regulation*
cm	centimètre (10^{-2} m)
CPU	*Central Processing Unit* – Processeur ou microprocesseur
ddl	Degré de liberté
dm	décamètre (10 m)
DSP	*Digital Signal Processing* – Processeur dédié au traitement de signal
Go	gigaoctets (10^9 octets) équivaut à 8 589 934 592 de bits
GPS	*Global Positioning System* – Système de positionnement global relatif au référentiel terrestre qui fonctionne par triangulation selon plusieurs balises (satellitaire et au sol)
He-Ne	Laser hélium-néon (635 nm)
HSV	*Hue-Saturation-Value* – Espace de couleur représenté par la teinte, la saturation et la valeur
IEEE	*Institute of Electrical and Electronics Engineers*
IEEE-1394	Norme de communication série à haut débit (souvent appelé *FireWire*)
INO	Institut National d'Optique

kg	kilogramme (10^3 g)
km	kilomètre (10^3 m)
K-NN	*K Nearest Neigbourhood* – technique des *k* plus proches voisins
MHz	megahertz (10^6 Hz)
MIL	*Matrox Imaging Library* – bibliothèque de programmation dédiée au traitement d'images
mm	millimètre (10^{-3} m)
Mo	megaoctets (10^6 octets) équivaut à 8 388 608 de bits
ms	milliseconde (10^{-3} s)
mW	milliwatt (10^{-3} W)
nm	nanomètre (10^{-9} m)
OpenGL	*Open Graphic Library* – Bibliothèque de programmation graphique 3D
RAM	*Random Access Memory* – mémoire vivre
RGB	*Red-Green-Blue* – Espace de couleur représenté par les composantes rouges. vertes et bleues
RVE	Représentation Virtuelle de l'Environnement
SE	Surface d'Élévation
UML	*Unified Modeling Langage* – Méthode de conception logicielle basée sur les techniques de programmation objet
USB	*Universal Serial Bus* – Lien de communication série à haut débit

INTRODUCTION GÉNÉRALE

Tenter de définir ce qu'est l'intelligence est un sujet délicat. Certaines personnes la définissent comme étant la faculté de comprendre ou de saisir par la pensée tandis que d'autres la décrivent comme étant la capacité de créer ou de concevoir par la manipulation de concepts abstraits. L'une des définitions les plus couramment utilisées est l'aptitude de s'adapter à une situation, de choisir en fonction des circonstances particulières et d'établir les liens causaux entre les différents phénomènes et évènements. Néanmoins, peu importe la définition précise que l'on donne à l'intelligence, l'homme est toujours fasciné par cette faculté qui se présente de façon si différente dans la nature. Que l'on parle de la forme aussi primaire que l'intelligence collective d'une société d'insectes ou de l'intelligence conceptuelle des êtres humains, l'homme a depuis très longtemps tenter d'en comprendre les mécanismes les plus subtils pour éventuellement en tirer profit.

L'adaptabilité de l'homme s'est particulièrement manifestée par son sens créatif et sa capacité à se doter d'outils appropriés en fonction de ses besoins. Même si à une époque, le silex a marqué de façon importante l'évolution du comportement humain, les outils modernes atteignent des niveaux de performance parfois impensables il y a seulement quelques dizaines d'années. Maintenant, dans sa quête constante de création, l'homme aborde l'un des défis les plus stimulants et les plus convoités depuis des générations : l'outil ultime.

Certains critères pouvant définir cet outil ultime sont les suivants : exécuter une tâche complexe de façon entièrement autonome, aider l'homme à faire tout type de travaux (qu'ils soient physiques ou intellectuels), s'adapter à différentes situations, apprendre rapidement et, à la limite, faire partie de la vie quotidienne des humains en tant que compagnon de vie. Même si ces derniers critères sortent tout droit de la science fiction, il n'en demeure pas moins que cet outil ultime correspond finalement à une entité

capable de se déplacer, apte à réaliser des travaux nécessitant certaines manipulations et, surtout, muni d'une *intelligence*.

Malgré les percées scientifiques et technologiques importantes des dernières années, ce type d'outil n'existe pas encore, même sous une forme primitive. Les robots actuellement disponibles sont pour la plupart dotés d'une intelligence extrêmement limitée. En fait, il serait plus juste de dire que dans la majorité des cas, ils réagissent en fonction de règles pré-établies qui définissent leur comportement et qui simulent leur adaptabilité. Heureusement, les recherches en cours montrent que certains résultats préliminaires sont à nos portes. D'ailleurs Moravec (2002) prédit qu'il y aura des robots destinés aux marchés de masse avant 2010 et des robots entièrement intelligents avant 2050!

Un des axes de recherche les plus importants en robotique est sans contredit la mobilité des robots. Les robots ne doivent plus être contraints sur un socle mais être libres de circuler librement et de réaliser ainsi un plus grand nombre de tâches. La mobilité des robots est un problème touchant principalement six domaines technologiques :

a. la perception de l'environnement;

b. la fusion des données, la représentation et l'interprétation de l'environnement;

c. la planification de trajectoire;

d. le positionnement;

e. le suivi de la trajectoire;

f. l'aspect mécanique et moteur du robot.

Malgré l'existence de plusieurs prototypes démontrant leur capacité mécanique et leur capacité de commande, il n'existe pas de robot qui intègre de façon élégante et parfaitement fonctionnelle tous les six aspects.

Le projet Capra est né du rêve de créer cet outil ultime. Capra, qui signifie chèvre en latin, est un robot quadrupède ayant une démarche dynamique et un comportement autonome. L'ambitieux projet que sont la conception et le développement de Capra se caractérise par l'aspect générique du projet. Le robot n'est pas destiné et optimisé pour une application particulière mais plutôt pour le maximum d'applications possibles. Les caractéristiques générales du robot sont les suivantes. Il est doté de quatre pattes ayant trois degrés de liberté chacune et un système de commande dynamique pour la démarche permettant de gérer une marche où il est en constant déséquilibre. Il peut transporter une charge pouvant atteindre 100 kg. Il est d'une taille lui permettant de se déplacer partout où un être humain peut aller et il peut se mouvoir de façon autonome dans son environnement selon des objectifs pré-établis. Pour une description plus exhaustive du projet, le lecteur peut se référer à Lambert (1999), à Lagarde (1999), à DeBlois, Lambert et Lessard (2000), à Lupien, Lessard, Demers (2001), à Lupien (2001) et à Lessard (2002).

Ce mémoire présente la conception et le développement de trois aspects importants du robot : la perception de l'environnement, la représentation virtuelle de l'environnement et la planification de trajectoire. Il est important de mentionner dès maintenant que l'élaboration d'une solution permettant de résoudre ces trois problèmes est une tâche colossale. Étant donné que ce mémoire présente les résultats préliminaires du projet de recherche, aucune des solutions présentées ici n'est développée en profondeur. On étudie plusieurs approches plutôt que de se limiter à l'optimisation d'une seule technique (notamment pour les deux dernières parties).

La figure 137 se trouvant à l'annexe 1 est un schéma de la structure conceptuelle de Capra. On voit les interconnexions qui existent entre les différents modules du robot. On remarque les trois couches principales : l'information, la décision et l'action. Les parties abordées dans ce mémoire sont des éléments des trois blocs ayant les cadres les plus larges sur la figure.

Problématique

Le problème se pose comme suit : trouver une solution élégante et fonctionnelle permettant à un robot marcheur de naviguer de façon autonome tout en profitant du plein potentiel qu'offre ce type de robot.

La solution doit considérer tous les types d'environnements terrestres : connus, partiellement connus ou totalement inconnus. Ce travail touche les problématiques associées à la perception de l'environnement, à la représentation virtuelle de l'environnement et à la planification de trajectoire.

Objectif global

L'objectif du projet est d'identifier et de valider certaines solutions technologiques permettant de résoudre les trois aspects de la problématique. La solution doit considérer principalement les capacités supplémentaires de déplacement que permet le robot marcheur.

Dans cette optique, la perception de l'environnement doit être en mesure de percevoir le relief au sol. La représentation virtuelle de l'environnement doit être en mesure d'identifier les obstacles se trouvant sur le terrain (on verra que la définition d'obstacle est décrite de façon précise pour un robot marcheur). Finalement, la planification de trajectoire doit déterminer un chemin valide pour que le robot puisse atteindre une nouvelle position spatiale dans l'environnement.

Objectifs spécifiques

1. Perception de l'environnement (Partie 1) :
   a. prendre des mesures précises du terrain se trouvant devant le robot;
   b. couvrir une surface suffisante pour que le robot puisse anticiper son comportement en fonction de l'environnement;
   c. faire un relevé relativement dense du terrain.

2. Représentation virtuelle de l'environnement (Partie 2) :
   a. fusionner et synthétiser l'information du module de perception de l'environnement;
   b. représenter le relief au sol;
   c. identifier les obstacles formés par :
      i. les pentes trop abruptes;
      ii. les discontinuités verticales présentes sur le terrain.
3. Planification de trajectoire (Partie 3) :
   a. déterminer une trajectoire globale pour que le robot passe de sa position courante à la position désirée;
   b. étudier différentes solutions alternatives.

Contraintes

a. La complexité de tout le processus doit être minimisée pour que tous les algorithmes puissent être implantés réellement sur un robot dont la puissance de calcul est limitée.
b. Le système de perception de l'environnement doit rester fonctionnel dans tous les types d'environnement et pour toutes les conditions environnementales (de jour comme de nuit). Il doit aussi permettre une détection de tous les éléments de la scène.
c. Les composantes physiques du système de perception de l'environnement doivent présenter un dimensionnement et un poids minimum. Ils ne doivent pas excéder la dimension de la tête qui est de 30 × 10 × 20 cm (pour la largeur, la hauteur et la profondeur respectivement).

Hypothèses de départ

a. La position et l'orientation du système de perception de l'environnement sont connues par rapport au robot.
b. La position et l'orientation du robot sont connues par rapport à l'environnement.

c. L'environnement est statique : aucun acteur de la scène n'est en mouvement.
d. Les erreurs de mesure du système de perception de l'environnement ne sont pas considérées dans les modules de représentation virtuelle de l'environnement et de la planification de trajectoire.

Méthodologie et approches préconisées

L'approche proposée présente une solution globale qui correspond bien aux objectifs, contraintes et hypothèses de départ. Par leur nature différente, chaque module nécessite des approches et des solutions technologiques différentes.

Perception de l'environnement (Partie 1)

La solution présentée pour ce premier module est reliée au développement d'un nouveau type de capteur : un système de vision stéréoscopique binoculaire avec lumière structurée. En fait, le capteur stéréoscopique utilise un projecteur laser muni d'un réseau de diffraction matriciel dans le but de rehausser le contraste de certains points au sol. Ce contraste permet de définir des références précises sur l'environnement. Les caméras, étant munies de filtres interférentiels de la longueur d'onde correspondant au laser, permettent de donner des images modifiées de la scène où apparaissent avec un grand contraste les points laser. De cette façon, il est facile pour le système de vision stéréoscopique de déterminer la position spatiale réelle des observations importantes des images que sont les images des points laser.

La solution nécessite un effort de développement important qui couvre la sélection des composantes, la disposition physique du système et tout le processus algorithme requis permettant de solutionner le problème de vision tridimensionnel (3D).

La solution algorithmique passe par cinq étapes importantes. De ces cinq étapes deux se démarquent par leur complexité et méritent d'être décrites dès maintenant.

a. le calibrage du système de vision;

b. l'acquisition des images;

c. l'extraction des observations se trouvant sur les images :

   i. la segmentation des images;

   ii. la localisation des observations;

   iii. le calcul des descripteurs de forme pour chaque observation;

   iv. la classification des observations;

d. l'appariement des observations;

   i. la rectification des observations;

   ii. l'identification des observations se trouvant sur les droites épipolaires conjuguées;

   iii. l'élaboration des groupes d'appariements possibles;

   iv. le calcul de la matrice de coûts pour chaque appariement possible;

   v. l'identification du meilleur groupe d'appariement

e. la reconstruction 3D.

La classification des observations (étape c-iv) est en fait un problème de reconnaissance de formes. La solution présentée pour ce problème particulier montre une analyse en composantes principales (permettant d'identifier les descripteurs de formes suffisant à décrire chaque observation) et une étude de trois types de classificateurs : le Bayes quadratique, les k plus proches voisins (*K-NN*) et le réseau de neurones de type perceptron multicouche. De plus, pour réaliser l'étude des observations, la construction d'une base de données contenant plus de 5 400 observations est réalisée.

Représentation virtuelle de l'environnement (Partie 2)

La représentation virtuelle de l'environnement sert d'interface entre le module de perception de l'environnement et les modules de planification de trajectoire et de contrôle de la démarche. Le rôle de cette interface est de contenir une représentation informatique viable pour les deux modules qui l'utilisent.

La solution présentée passe par six étapes :

a. l'insertion des données provenant du module de perception de l'environnement;

b. la représentation sous forme de nuage de points;

c. la représentation sous forme de surface d'élévation;

d. la représentation des obstacles correspondants aux discontinuités verticales;

e. la représentation des obstacles correspondants aux pentes trop abruptes;

f. la synthèse des différentes représentations.

Planification de trajectoire (Partie 3)

Cette troisième partie présente les méthodes permettant de déterminer une trajectoire suivie par le robot pour se rende d'un point A à un point B sans heurter un obstacle. Le problème de définition des trajectoires des pattes est laissé au module de contrôle de la démarche. Ce module doit coordonner les pattes pour mouvoir le robot convenablement dans le but d'atteindre les objectifs de déplacement global déterminés par le module de planification de trajectoire.

On présente ici une approche intéressante consistant à déterminer le chemin le plus court suivant les contours d'obstacles polygonaux. Cette méthode consiste à décrire les obstacles comme étant des polygones convexes et à trouver le chemin le plus court parmi tous les chemins possibles. Les chemins sont définis par des segments de droites

reliés par les sommets tangents des obstacles. Cet algorithme est constitué de huit étapes :

a. formation des polygones décrivant les obstacles;

b. réduction du nombre de sommets décrivant les polygones;

c. création de polygones convexes;

d. croissance des polygones;

e. identification des chemins possibles;

f. réduction du nombre de chemins par les chemins tangents aux obstacles

g. calcul des coûts associés aux différents segments de chemin;

h. recherche du chemin le plus court.

Structure particulière du document

Comme on vient de le constater, le travail de recherche porte sur trois aspects différents et complémentaires. Ce mémoire est donc divisé en trois parties distinctes afin de faciliter la compréhension de chaque étape du processus dans le contexte global du projet. La première partie présente le module de perception de l'environnement. Les chapitres 1 à 5 couvrent tous les aspects de conception et de développement du capteur. La deuxième partie expose le module de représentation virtuelle de l'environnement. Cette partie est couverte par les chapitres 6 et 7. La troisième partie présente le module de planification de trajectoire ainsi qu'un simulateur. Les chapitres 8 et 9 montrent une solution viable à la problématique de planification de trajectoire. Finalement, le chapitre 10 présente le simulateur global du projet ainsi que plusieurs résultats obtenus.

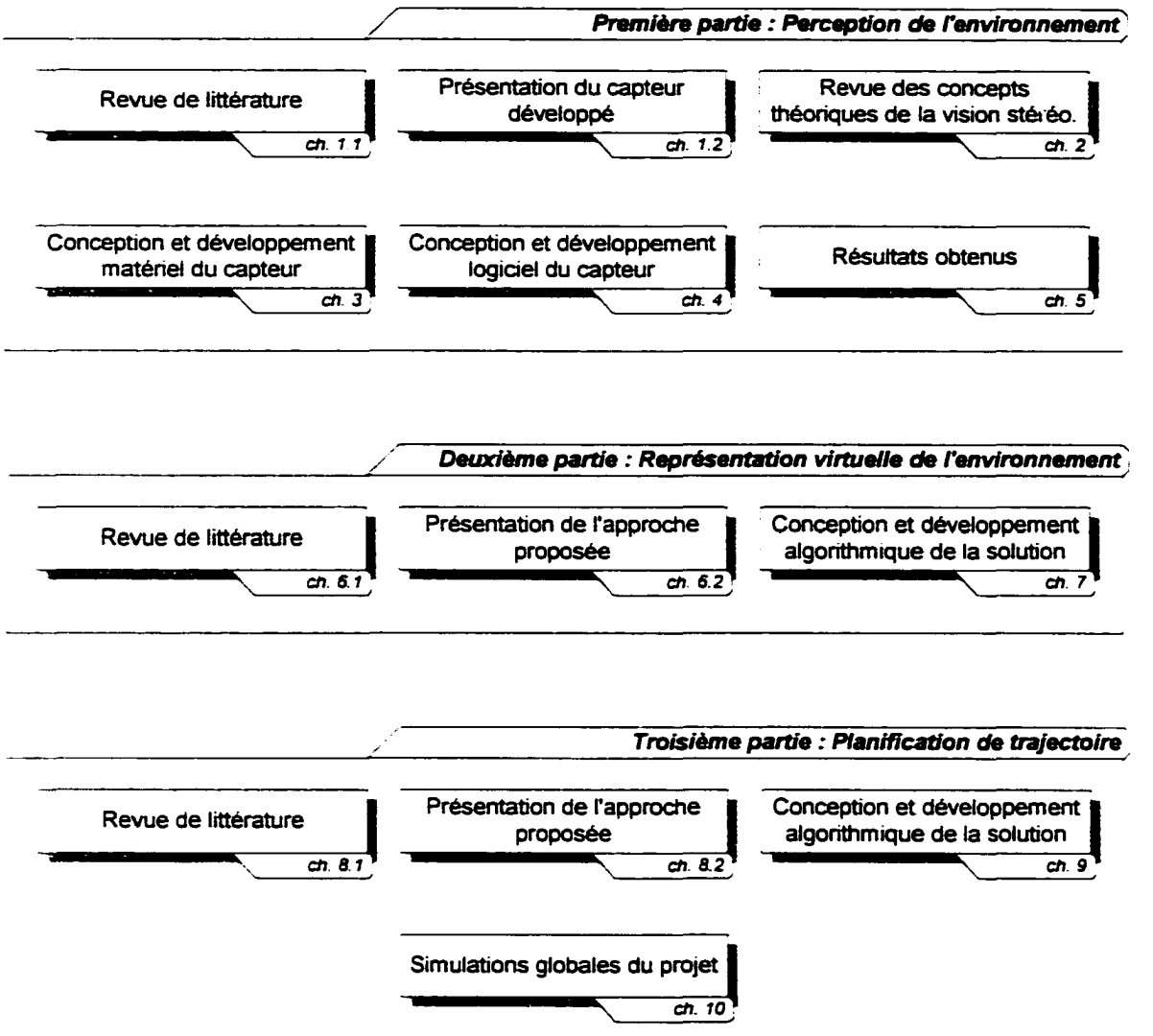


Figure 1 : Présentation des parties du document

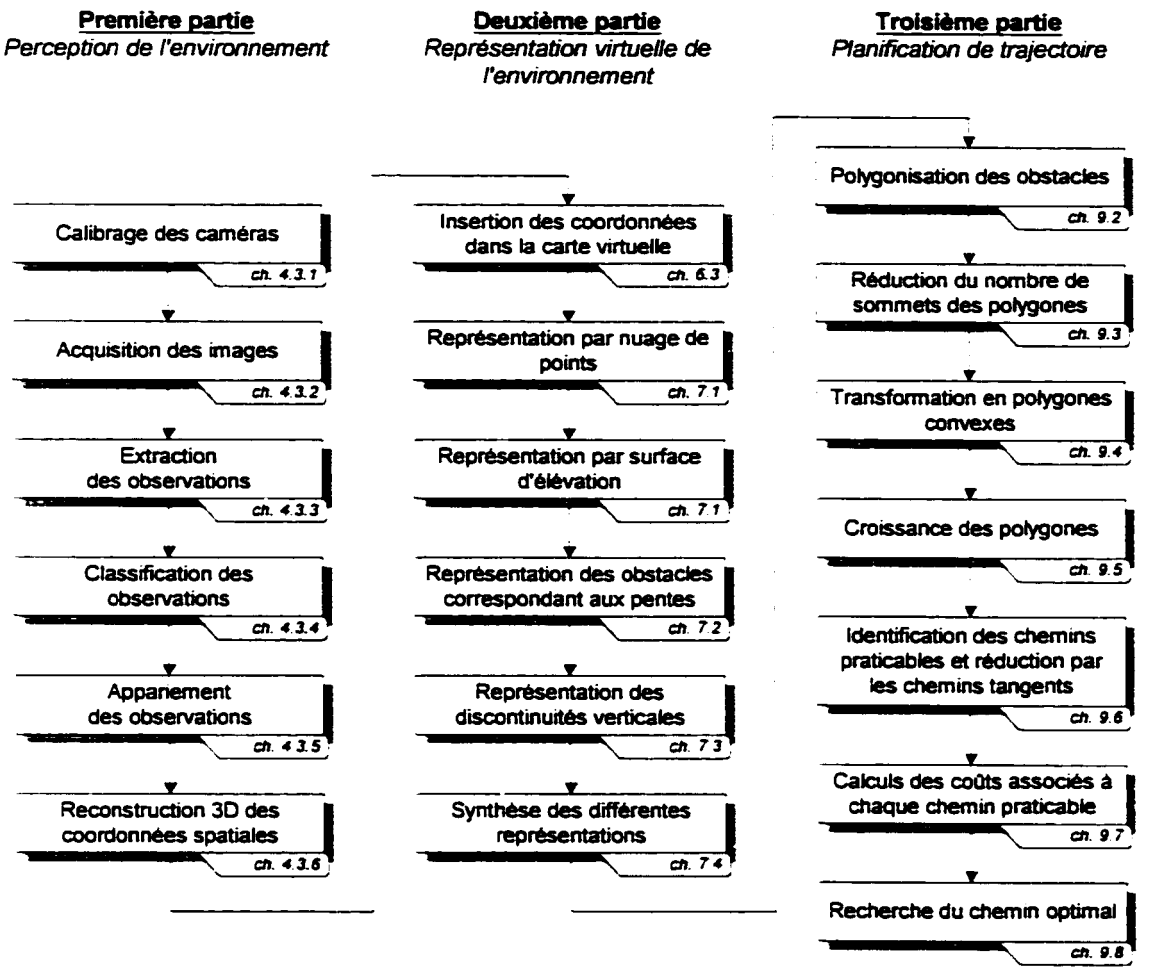


Figure 2 : Structure algorithmique complète

PREMIÈRE PARTIE

PERCEPTION DE L'ENVIRONNEMENT

INTRODUCTION

À la base d'un processus décisionnel, l'information disponible est l'élément le plus important. La pertinence d'une décision vient d'abord et avant tout de la qualité des informations à l'entrée du système. Dans le cadre d'une application en robotique mobile, et encore plus lorsque le robot doit évoluer de façon autonome, l'acquisition et l'interprétation de l'information sont des problèmes complexes nécessitant un effort considérable de développement et d'intégration.

Les capacités supplémentaires du robot marcheur par rapport aux autres robots terrestres viennent principalement de son aptitude à se mouvoir sur des terrains plus accidentés. Par exemple, un robot sur roues ou à chenilles ne peut pas franchir un obstacle surélevé dont la taille est supérieure à la moitié de la dimension des roues. Par contre, un robot sur pattes peut, s'il est assez agile, se propulser pour grimper et franchir cet obstacle. Évidemment, cette comparaison ne s'applique qu'aux robots terrestres et elle ne peut pas s'appliquer aux robots aquatiques ou aériens.

Pour être efficace, le robot marcheur doit connaître les endroits où il peut déposer ses pieds. Même si le robot possède l'intelligence et les réflexes d'ajuster son comportement et sa démarche sur un terrain complètement inconnu, il est préférable qu'il puisse prévoir à l'avance les endroits fiables sur le terrain : ainsi il n'aura pas à réagir inutilement. D'ailleurs, Dudek et Jenkin (2000) soutiennent que les performances d'un robot marcheur peuvent être grandement améliorées par l'utilisation de cette connaissance supplémentaire sur l'environnement.

Suite à ces réflexions relatives aux robots marcheurs, connaître la nature du terrain sur lequel évolue le robot est essentiel pour que ce dernier puisse profiter pleinement des avantages qu'il offre. Idéalement, il ne suffit pas de connaître l'aspect géométrique du sol mais aussi la composition de celui-ci. Ce mémoire n'aborde pas cette dernière problématique et se concentre seulement sur l'aspect géométrique du terrain.

Néanmoins, l'information géométrique reste la plus importante et elle permet de tirer profit des avantages importants qu'offre le robot marcheur.

Pour qu'un système de planification de trajectoire soit efficace, la représentation virtuelle de l'environnement doit être fidèle à la réalité et surtout elle doit représenter le maximum de contraintes existantes. Or si on désire une application générique permettant au robot de naviguer de façon autonome dans tous les environnements possibles, celui-ci doit pouvoir percevoir tous les types d'obstacles. On verra, dans la deuxième partie du mémoire, que les obstacles contournés par un robot marcheur diffèrent de la simple définition habituelle. En effet, les contraintes de déplacement peuvent être résumées par trois situations possibles : les pentes ainsi que les discontinuités verticales et horizontales. Donc, en plus de percevoir les obstacles à moyenne et longue portée (comme les murs et les arbres), il lui est essentiel de connaître le relief du terrain.

Différents types de technologies permettent de percevoir l'environnement et sont déjà des solutions fiables et éprouvées autant dans le domaine industriel que scientifique. Certains offrent des performances allant au delà des capacités de perception des êtres vivants. Il existe aussi de plus en plus de capteurs extéroceptifs développés et commercialisés spécifiquement pour la robotique mobile. Ces nouveaux capteurs offrent souvent une grande robustesse, une consommation énergétique moindre ainsi qu'un encombrement et un poids réduits. Certains permettent une mesure par contact alors que d'autres peuvent mesurer à courte, moyenne ou longue portée. De plus, certains capteurs offrent une précision remarquable. En fait, il existe tellement de technologies disponibles actuellement que le premier défi est de déterminer quels produits utiliser ou quelles technologies mettre à profit pour développer un capteur extéroceptif. Évidemment, un tel développement doit se faire en pleine connaissance des capacités et des limites des technologies utilisées. Bien que ces technologies soient limitées par leur capacité de perception ou le niveau de complexité d'intégration sur un

véhicule mobile, elles possèdent toutes des avantages et des inconvénients. Dans le but d'augmenter la gamme des capteurs extéroceptifs disponibles pour le robot, un système de vision stéréoscopique binoculaire avec lumière structurée est développé. Ce capteur est une solution élégante à un problème spécifique : réaliser le relevé topographique du sol se trouvant devant le robot. La première partie de ce mémoire présente ce nouveau capteur.

Ainsi cette première partie du mémoire est divisée en cinq chapitres et couvre la conception et le développement complet du capteur spécifié. Le chapitre 1 présente une brève revue de littérature couvrant les différentes techniques de perception tout en mettant l'emphase sur les technologies répondant bien au besoin spécifique qui nous occupe. On enchaîne par une définition formelle du problème pour ensuite présenter le capteur développé. Ce chapitre se termine par une revue exhaustive des avantages et inconvénients de ce capteur. Puisque ce dernier est basé sur le principe de la vision stéréoscopique, le chapitre 2 présente un bref rappel des concepts théoriques de la vision stéréoscopique. Le chapitre 3 présente l'aspect matériel du capteur, soit la sélection des composantes principales et la disposition physique du capteur sur le robot. Le chapitre 4 couvre l'aspect principal de la conception et du développement, c'est-à-dire l'aspect algorithmique. Finalement, le chapitre 5 présente les résultats préliminaires du prototype.

CHAPITRE 1

DÉFINITION DE LA PROBLÉMATIQUE

1.1 Revue de littérature

Il convient d'identifier quel type de capteur permet de modéliser en trois dimensions le terrain se trouvant devant le robot. Même si la tâche semble simple, on se rend rapidement compte que le travail à faire est colossal et complexe si on considère les cinq critères suivants :

1. Le capteur extéroceptif doit être embarqué sur le robot, ce qui impose des contraintes importantes pour le poids, l'encombrement et la quantité de données à traiter (cette dernière contrainte étant imposée par la puissance de calcul limitée à bord du robot).

2. Le capteur étant fixé sur le robot, il doit avoir une certaine résistance aux chocs et aux vibrations. Cette résistance doit être non seulement au niveau de l'intégrité physique du capteur mais aussi de la précision des mesures. Cette résistance est d'autant plus importante que les mouvements complexes et aléatoires de va et vient d'un marcheur peuvent créer des variations instantanées sur la position et l'orientation désirées du robot. D'ailleurs, ces mouvements sont souvent de très grande amplitude autant en position, en vitesse et en accélération. De plus, on doit prévoir un système de position et d'orientation relatives pour permettre la correction des mesures prises sur le terrain.

3. Le capteur doit couvrir une surface du terrain suffisamment grande se trouvant devant le robot. Ce besoin vient de la dynamique du déplacement du robot car il doit percevoir assez tôt les obstacles l'obligeant à s'immobiliser ou à changer de trajectoire. La dimension de cette surface est fonction de la vitesse maximale du robot, de sa capacité à s'arrêter et du temps de réaction du processus décisionnel (acquisition, traitement des données, décision et action).

4. Le capteur doit faire un échantillonnage relativement dense du terrain. Ce critère est évident si on considère des obstacles aussi étroits qu'un poteau de 10 cm par exemple. Évidemment, cette dimension minimum est fonction de la nature du terrain et de la dimension des obstacles que le robot doit contourner. Lorsque cette dimension minimum des obstacles est déterminée, il faut utiliser un capteur étant en mesure de faire un échantillonnage d'au moins la moitié de la dimension requise (ce qui permet de respecter le critère de Nyquist sur les fréquences d'échantillonnage).

5. Le capteur doit être en mesure de donner des informations très rapidement. Non seulement le temps d'exposition doit être considéré comme instantané (à cause des mouvements de grande amplitude du marcheur) mais aussi les calculs requis doivent être assez brefs pour permettre une compilation rapide des données. Ceci permet de bien comprendre l'environnement et de prendre les bonnes décisions. Cette vitesse d'acquisition est fonction de la vitesse du robot et du traitement des algorithmes qui suivent le système de perception : la fusion des données, l'interprétation de la carte virtuelle, la planification de trajectoire ainsi que le suivi de trajectoire.

Une revue exhaustive de tous les types de capteurs ne trouve pas sa place ici. D'ailleurs il existe un ouvrage sur ce sujet qui le couvre de façon remarquable. Everett (1995) présente tous les aspects théoriques et les applications réalisées par toutes les technologies actuellement disponibles. Chavand et Colle (1998) abordent plusieurs

concepts mathématiques associés à la perception de l'environnement (comme les erreurs de mesures, la fusion d'informations, les capteurs et la localisation). Une première analyse permet de rejeter plusieurs types de capteurs et de favoriser un capteur sans contact pouvant faire une mesure ponctuelle de l'environnement. Il existe principalement trois solutions qui répondent bien à ces cinq critères.

i. La première solution est un capteur de type télémètre capable de mesurer une tranche du terrain par balayage. Ce type de capteur est basé sur l'un des deux principes suivants : la triangulation avec une source lumineuse ou le temps de vol d'une onde (ultrasonique ou lumineuse). Par exemple, la compagnie allemande Sick fabrique un modèle de télémètre couramment utilisé en robotique. Le modèle LMS 291-S05 OUTDOOR[1] fait un balayage de 180°, possède une portée qui va jusqu'à 80 m et a une précision de 10 mm.

Figure 3 : Télémètre LMS 291-S05 OUTDOOR de Sick™

Toutes ces caractéristiques sont amplement suffisantes pour les besoins identifiés pour le robot marcheur. Il existe par contre deux désavantages importants à ce capteur. En premier lieu, le poids et la dimension sont vraiment inadéquats pour un robot marcheur parce que ces caractéristiques sont des critères de premier ordre (4,5 kg – 35×19×27 cm). Évidemment, il est possible

[1] Voir : www.sick.com/de/en/products/categories/auto/lasermeasurementsystemsoutdoor/lms291s05outdoor.html

de développer soi-même un tel capteur et de tenter de réduire suffisamment la taille et le poids. Malgré les efforts importants qu'engendre le développement d'un tel capteur, le principe même de ce dernier est relié à un inconvénient important : même si ce capteur offre une excellente résolution spatiale dans l'axe de balayage (distance entre deux mesures), il n'en reste pas moins que le capteur se limite à un seul axe. C'est comme regarder le sol à travers une fente horizontale faite dans un masque. Il est possible de créer un capteur qui balaye une surface complète au lieu d'un axe mais les efforts de développement sont considérables compte tenu de la complexité qu'exigent les performances demandées.

ii. La deuxième solution est un système de vision utilisant une ligne laser qui découpe et rehausse le relief du sol. Il existe plusieurs produits sur le marché qui fonctionnent sur ce principe. Par exemple, la compagnie ServoRobot[2] développe et vend une gamme importante de caméra laser. Récemment, l'Institut National d'Optique (INO) a développé le MapScan[3] qui fonctionne également sur ce principe.

Figure 4 : Caméra laser MapScan™ développé par l'INO

[2] Voir : www.servorobot.com/fieldsofview.htm
[3] Voir : www.ino.ca/fr/syst_et_compo/3dms.asp

Malheureusement, il ne semble pas exister de produits commercialement disponibles répondant aux besoins identifiés. Les plages de mesure des outils déjà existants sont beaucoup trop réduites pour les besoins d'un robot marcheur. Malgré tout, le principe reste simple et le développement d'un tel capteur est facilement réalisable. Néanmoins, il existe au moins trois inconvénients pour l'usage d'un tel capteur. 1) L'analyse ne se fait que sur un plan, le plan issu de la ligne laser. 2) Un tel système requiert un laser puissant capable de générer une image claire. L'utilisation d'un laser puissant génère un problème de poids, de dimension et surtout de sécurité. 3) Selon Fournier (1999), la solution reste simple dans le cas où le capteur reste toujours dans la même position et dans la même orientation par rapport au sol. Mais dès que l'orientation du système varie, la complexité des algorithmes de correction rend le développement beaucoup plus complexe.

iii. La troisième solution consiste en un système de vision artificielle capable d'évaluer la profondeur. Malgré le succès des techniques de vision 3D basées sur le mouvement ou les ombrages (voir Trucco et Verri 1998), ces méthodes sont complexes, difficiles à implanter et relativement peu performantes comparativement à un système de vision stéréoscopique. Néanmoins, peu importe le système de vision développé, la solution est complexe étant donné le caractère générique du projet. Autrement dit, le système de vision doit rester performant peu importe la nature des images se trouvant à l'entrée. Les algorithmes actuels de vision ne permettent pas un traitement rapide et infaillible sur n'importe quel type d'image. Moravec (2002) montre qu'il a développé un système de vision stéréoscopique trinoculaire très sophistiqué permettant une compréhension de l'environnement de très haut niveau. Malheureusement, son approche nécessite une trop grande puissance de calcul pour les ordinateurs actuels. Il prétend que d'ici quelques années, les ordinateurs auront atteint une puissance de calcul suffisante pour utiliser en

temps réel la technologie qu'il a développée. Ainsi même pour un système de vision stéréoscopique, binoculaire ou trinoculaire, le développement reste complexe et peut-être même irréaliste avec les outils actuellement disponibles dans le cadre des objectifs définis. La compagnie Point Grey Research a développé et vend un produit nommé Digiclops[4] qui est exactement un système de vision stéréoscopique trinoculaire. Malgré les résultats surprenants du Digiclops, les mêmes limitations s'appliquent à ce produit.

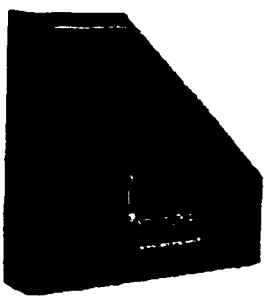


Figure 5 : Système de vision stéréoscopique trinoculaire Digiclops™ de Point Grey Research™

Le capteur qui est présenté ici est un amalgame des solutions précédentes. C'est un système de vision stéréoscopique binoculaire avec lumière structurée. Au système de vision stéréoscopique standard, on ajoute un laser projetant une matrice de points, le tout étant orienté vers le sol. La stéréoscopie permet de retrouver la troisième dimension et l'utilisation du laser dirigé vers le sol génère des marques de référence faciles à interpréter dans n'importe quelle condition. Évidemment, ce capteur présente aussi ses avantages et ses inconvénients.

[4] Voir : www.ptgrey.com/products/digiclops/index.htm

1.2 Approche proposée

On propose un nouveau capteur extéroceptif de haut niveau qui est un système de vision stéréoscopique binoculaire actif[5]. Tel que mentionné par Everett (1995), un tel système a été développé pour le robot ROBART II[6]. Ce premier système est plus simple car il est constitué de capteurs d'images linéaires et la source lumineuse est une lampe incandescente projetant une forme en V. Évidemment les moyens technologiques restreints de l'époque ont limité le développement de ce capteur.

Le concept du capteur présenté est simple : on utilise un système de vision stéréoscopique pour déterminer la profondeur des éléments de l'environnement imagés sur les caméras. Celles-ci sont munies de filtres interférentiels ne permettant de voir que la matrice de points laser projetés au sol. On verra que cette simplification a des répercussions importantes sur tout le processus. D'ailleurs la section suivante fait une revue exhaustive des avantages et des inconvénients de ce système.

Ce système de vision jumelé au déplacement du robot permet de donner un échantillonnage assez dense de la scène. La nature ponctuelle de ce qui est observé donne une représentation de l'environnement sous forme de nuage de points. Malgré la simplicité du concept, le cadre d'application générique du projet définit un ensemble de caractéristiques exigeantes pour la conception et pour le développement de ce capteur. C'est la simplification des images causée par le laser qui réduit suffisamment les algorithmes du système pour le rendre fonctionnel avec la technologie existante aujourd'hui. Pour être plus précis, le rôle de la matrice de points laser au sol est de

---

[5] Le terme *vision stéréoscopique actif* est utilisé pour deux types d'application dans la littérature : soit un système utilisant un éclairage artificiel (telle qu'une source de lumière structurée) ou soit un système stéréoscopique où chaque caméra peut pivoter indépendamment de l'autre (permettant ainsi d'augmenter la disparité et le champ de vision global du système). Dans ce mémoire, ce terme fait référence à un système de vision stéréoscopique projetant une lumière structurée.

[6] Voir : www.spawar.navy.mil/robots/land/robart/robart.html

former des points de références se retrouvant sur les images des caméras. Puisque celles-ci sont munies de filtres interférentiels, les images obtenues de la scène sont principalement composées d'observations claires représentant les points laser sur un fond plus sombre. La simplicité de ces images permet à des algorithmes relativement peu complexes d'évaluer la position spatiale de chaque point observé. Malgré tout, la réalisation complète d'un capteur de cette complexité nécessite un effort de développement non négligeable. En plus de considérer les besoins complexes inhérents à l'application courante, il faut tenir compte des technologies et des ressources disponibles (ressources matérielles, financières et humaines).

Cette première partie du mémoire présente tous les aspects de la conception et du développement du capteur. La partie principale du développement se situe au niveau de la solution algorithmique. L'approche algorithmique présentée utilise plusieurs techniques complémentaires permettant de converger vers une solution fonctionnelle et efficace. Les techniques de vision stéréoscopique, de traitement d'images, de reconnaissance de forme et d'analyse de données sont toutes mises à profit pour élaborer la solution. La partie matérielle du prototype est aussi abordée. On couvre la méthode de sélection et les caractéristiques importantes des principaux constituants. Le problème de sécurité inhérent à l'utilisation de la lumière laser est aussi présenté.

Les principaux constituants du capteur sont :

a. deux caméras munies de filtres interférentiels;
b. un laser muni d'un réseau de diffraction projetant une matrice de points au sol;
c. une unité de calcul;
d. une source d'alimentation;
e. un support mécanique.

Les figures 6 à 8 illustrent le principe du capteur développé. La première figure montre les liens existant entre les différents composants tandis que les images suivantes montrent des vues différentes des composants optiques.

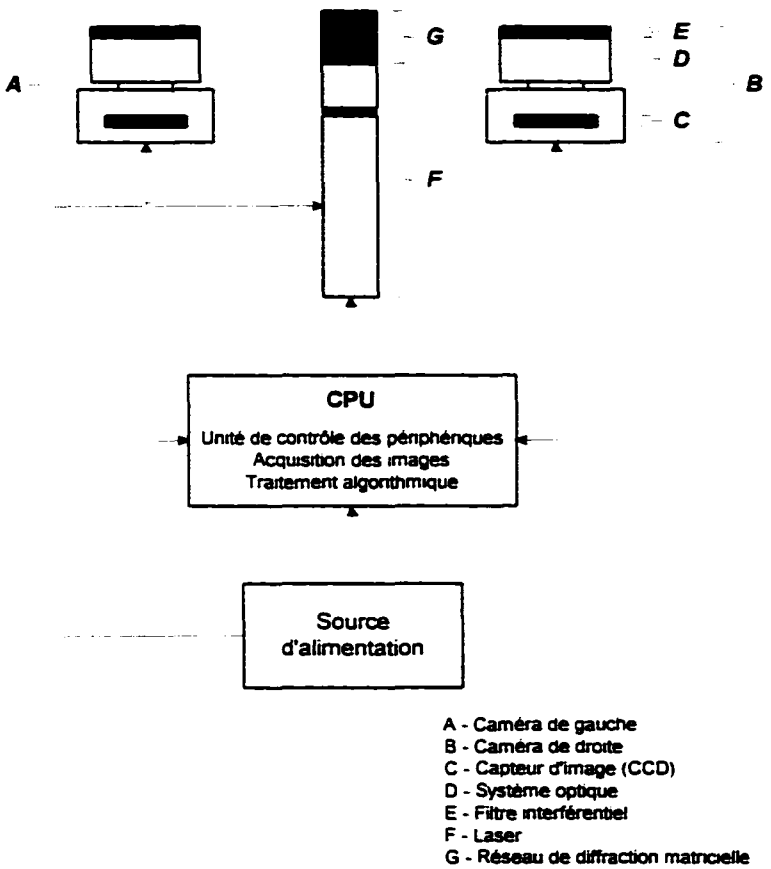


Figure 6 : Liens entre les composants du capteur développé

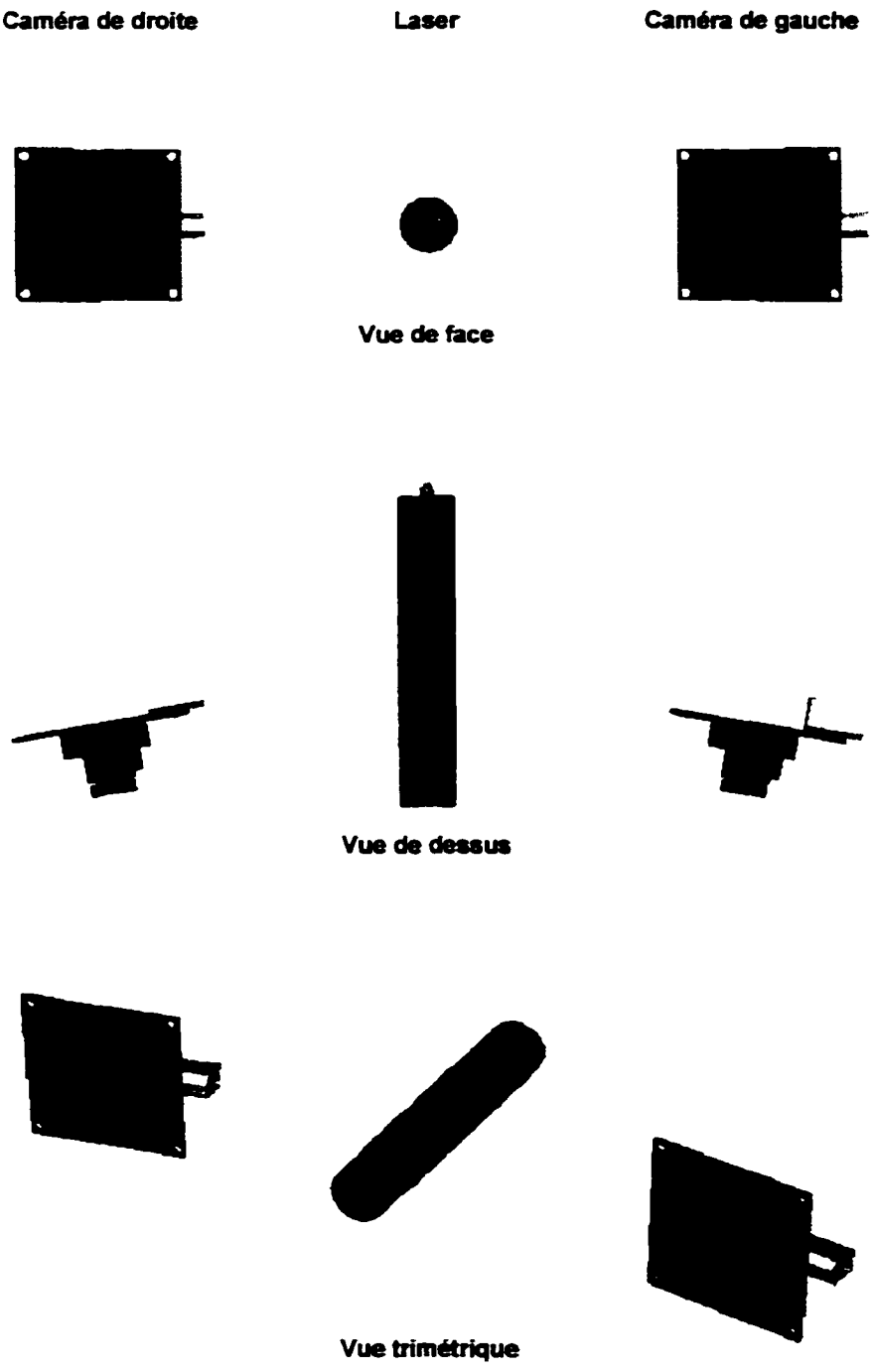
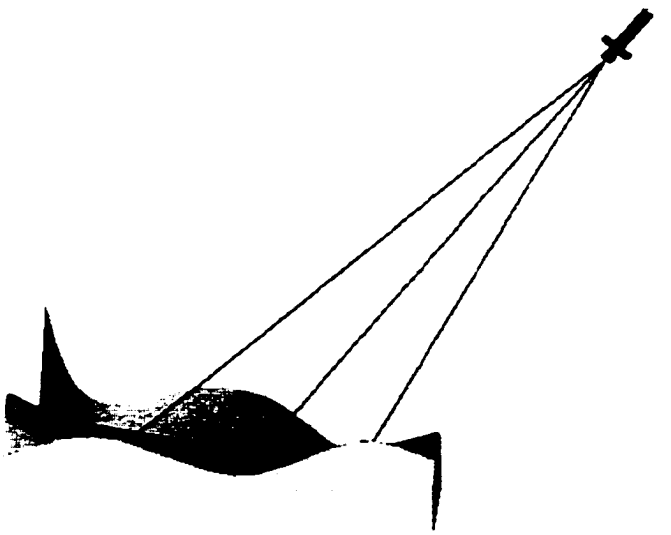
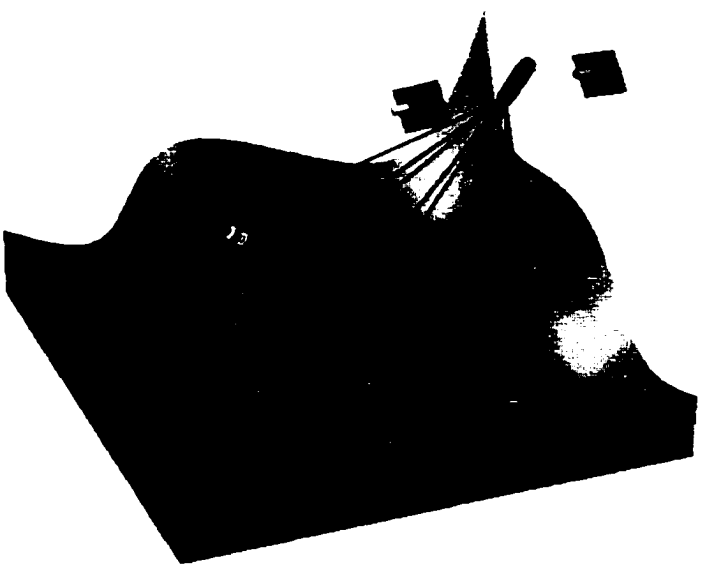


Figure 7 : Schéma du capteur développé (caméras et laser)



Vue de côté



Vue isométrique arrière

Figure 8 : Schéma du capteur développé (caméras, laser et projection des faisceaux laser sur une scène)

1.3 Avantages et inconvénients du capteur développé

Avant d'énumérer les avantages et les inconvénients du capteur, il est important de comprendre quelle est la conséquence la plus directe de l'utilisation d'un laser sur un système de vision stéréoscopique. Le fait de voir seulement quelques points de l'environnement limite considérablement ce qu'il est possible de percevoir du terrain. Ainsi cette restriction de la perception engendre une limitation de ce qu'il est possible d'interpréter du terrain et évidemment le comportement du robot en est affecté.

Malgré tout, certains avantages du capteur sont importants : le poids, les dimensions réduites, la capacité de résolution spatiale et la précision des mesures. D'autres avantages sont encore plus importants tels que la quasi-indépendance du système par rapport à la complexité de la scène et aux conditions d'éclairage. D'un autre côté, il existe deux inconvénients majeurs au capteur : la sécurité inhérente à l'utilisation d'une lumière laser et surtout la problématique reliée à la réflexion des points laser.

Malgré les avantages importants énumérés, les désavantages sont suffisamment importants pour remettre en question l'utilisation du système dans certains contextes et dans certaines conditions.

1.3.1 Limitation de la capacité de perception du capteur

Ce qui limite la capacité de perception du capteur est la nature même des observations puisque la scène est en fait une scène modifiée par un éclairage structuré. Le fait de munir les caméras de filtres interférentiels permet de voir seulement la lumière du laser et donne des images très différentes de la scène réelle. Ces dernières sont constituées de points laser et de quelques parasites provenant de l'éclairage ambiant sur un fond sombre. Ainsi en considérant deux images idéales (c'est-à-dire sans bruit), les seules observations présentes de la scène sont les images des points laser.

Figure 9 : Paire d'images stéréoscopiques sans bruit

L'analyse de ces images particulières permet seulement de retrouver des points de l'espace. Ainsi la première représentation (ou modélisation) résultant de l'acquisition d'informations par ce capteur est une description de l'environnement sous forme de nuage de points. Ce type de modélisation (sous forme de nuage de points) a des conséquences importantes sur l'interprétation de l'environnement et évidemment sur tout le processus de décision du robot. La compréhension géométrique de l'environnement est basée sur un ensemble de données éparses et non synthétisées. Les données brutes provenant du système de vision ne peuvent servir directement à la planification de la trajectoire. En fait, un traitement des données est essentiel pour réussir à identifier toutes les caractéristiques nécessaires à une compréhension suffisante de l'environnement. Il ne faut pas perdre de vue que ce nuage de points décrit toutes les caractéristiques de l'environnement sans aucune distinction. Ainsi aucune information spécifique ne permet de discerner les différents acteurs de la scène. Par exemple, le sol et les obstacles font partie intrinsèque des données. D'où l'importance d'une analyse subséquente sur les données permettant une compréhension adéquate de la représentation spatiale de l'environnement (la deuxième partie du mémoire couvre ce sujet).

De plus, l'utilisation d'un système de vision basé sur un système d'éclairage structuré possède quelques autres limitations de perception qui deviennent des désavantages

importants. L'intensité lumineuse d'un faisceau laser réfléchi sur la scène dépend de l'angle que fait la normale à la surface avec le laser mais aussi de la couleur de la surface. Ainsi, si la surface observée est noire, l'énergie de la lumière laser est absorbée et non réfléchie, ce qui implique qu'il est difficile de voir les surfaces noires. Un autre point important à considérer est la performance du système à l'extérieur lorsqu'il y a des précipitations (pluie, neige ou grêle). Les performances du système de vision n'ont pas été évaluées dans de telles conditions mais il est facile de penser que les nombreuses réflexions sur la pluie par exemple fausseront les données.

1.3.2 Quasi-indépendance du capteur aux conditions d'éclairage

La quasi-indépendance du capteur par rapport aux conditions d'éclairage vient aussi influencer grandement la capacité de perception du système. Tel que mentionné auparavant, si le système fait l'acquisition d'images idéales (sans bruit), alors seuls les points laser sont présents sur les images. Malheureusement, les images ont toujours un certain niveau de bruit dû à l'éclairage ambiant de la scène.

Si la lumière ambiante est monochromatique et différente de la longueur d'onde du laser ou polychromatique sans contenir la longueur d'onde du laser, alors, et uniquement dans ces cas, il est possible d'affirmer que le système est complètement indépendant des conditions d'éclairage. Dans tous les autres cas, la lumière ambiante a un effet plus ou moins important sur la qualité des images. Ainsi la lumière ambiante influence la qualité des images de façon proportionnelle à la puissance de la source lumineuse, à l'angle d'incidence de cette dernière par rapport aux caméras et aux angles de réflexion des éléments de la scène.

De plus, puisque les lasers ont une puissance maximum limitée, il faut que la lumière ambiante soit moins intense que celle du laser. Sinon les images risquent d'être saturées en intensité et dans ce cas le robot devient pratiquement *aveugle*.

Par contre, l'usage de lumière structurée pour éclairer la scène permet de voir même si aucune source de lumière n'est présente. Donc, dans la noirceur la plus totale, le robot est encore capable de se diriger.

1.3.3 Quasi-indépendance du capteur à la complexité de la scène

On peut définir la complexité de l'image d'une scène selon le nombre d'observations que cette dernière possède. Par exemple, l'image du sol d'un stationnement pour voitures est beaucoup moins complexe que l'image du sol d'une forêt en automne. Ainsi puisque seuls les points laser de l'image sont observés, il est possible d'affirmer que le processus de vision est pratiquement indépendant de la complexité de la scène, du moins dans une certaine mesure. La complexité des images reste toujours fonction de la complexité géométrique de la scène observée. Ainsi certains phénomènes d'occlusion, de changement brusque de profondeur ou de réflexions multiples peuvent engendrer un certain niveau de complexité mais tous ces phénomènes géométriques sont pratiquement insignifiants si on les compare aux problèmes d'éclairage, d'ombrage, de texture, de couleur, etc.

1.3.4 Réduction de la complexité des algorithmes

Une conséquence directe de ces quasi-indépendances est la simplification considérable des images et, par le fait même, du traitement algorithmique requis. On retrouve à l'entrée du processus analytique des images principalement constituées de points laser. De ces images, seulement les points laser sont extraits et analysés malgré la présence potentielle de parasites. Ainsi ces points sont la seule et unique source d'information de tout le processus algorithmique. En partant de l'acquisition d'image jusqu'à la reconstruction d'une représentation de l'environnement utilisable par le module de planification de la trajectoire, seulement des informations sous forme de points sont manipulées.

Le fait d'avoir seulement des points comme informations permet une optimisation pour chacun des algorithmes du processus.

1.3.5 Problématique reliée à la réflexion des points laser

Tous les faisceaux laser sont réfléchis sur la surface où ils sont projetés. Selon la nature de cette surface réfléchissante et des surfaces se trouvant à proximité, les réflexions secondaires du laser peuvent occasionner quelques fausses représentations spatiales.

Pour comprendre la nature du problème, on prend un faisceau laser et on regarde ce qui se passe lorsqu'il est réfléchi sur une première surface. Le point de la scène où se trouve le point laser ne dépend pas de la position de l'observateur. Ainsi l'image du point laser indique un seul et même endroit dans le monde réel 3D. Lorsque la réflexion diffuse est réfléchie sur une seconde surface suffisamment réfléchissante, la position de la deuxième réflexion du point laser dépend de la position de l'observateur. Autrement dit, deux observateurs placés à deux endroits différents voient l'image de ce point laser à deux endroits différents de la scène. Étant donné que le principe même de la vision stéréoscopique implique des images prises d'une même scène selon deux positions différentes, ce problème est constamment présent.

La figure de la page suivante illustre très bien cette problématique. Le premier schéma (en haut à gauche) montre que la première réflexion du point laser correspond à un seul endroit de la scène indépendamment de la position de l'observateur. Sur le deuxième schéma (en bas à gauche), on peut voir que même une réflexion secondaire issue d'une réflexion spéculaire génère un point laser sur un seul endroit de la scène. Il n'existe donc pas de problème pour ce type de réflexion. Par contre, sur le dernier schéma (en bas à droite), on peut voir que les réflexions secondaires issues d'une réflexion diffuse sont localisées à deux endroits différents selon le point de vue. Si les deux points images sont considérés comme étant ceux d'un seul élément de la scène, il y a une erreur

de reconstruction 3D. Ceci a pour conséquence d'ajouter un point virtuel qui n'existe pas dans le nuage de points.

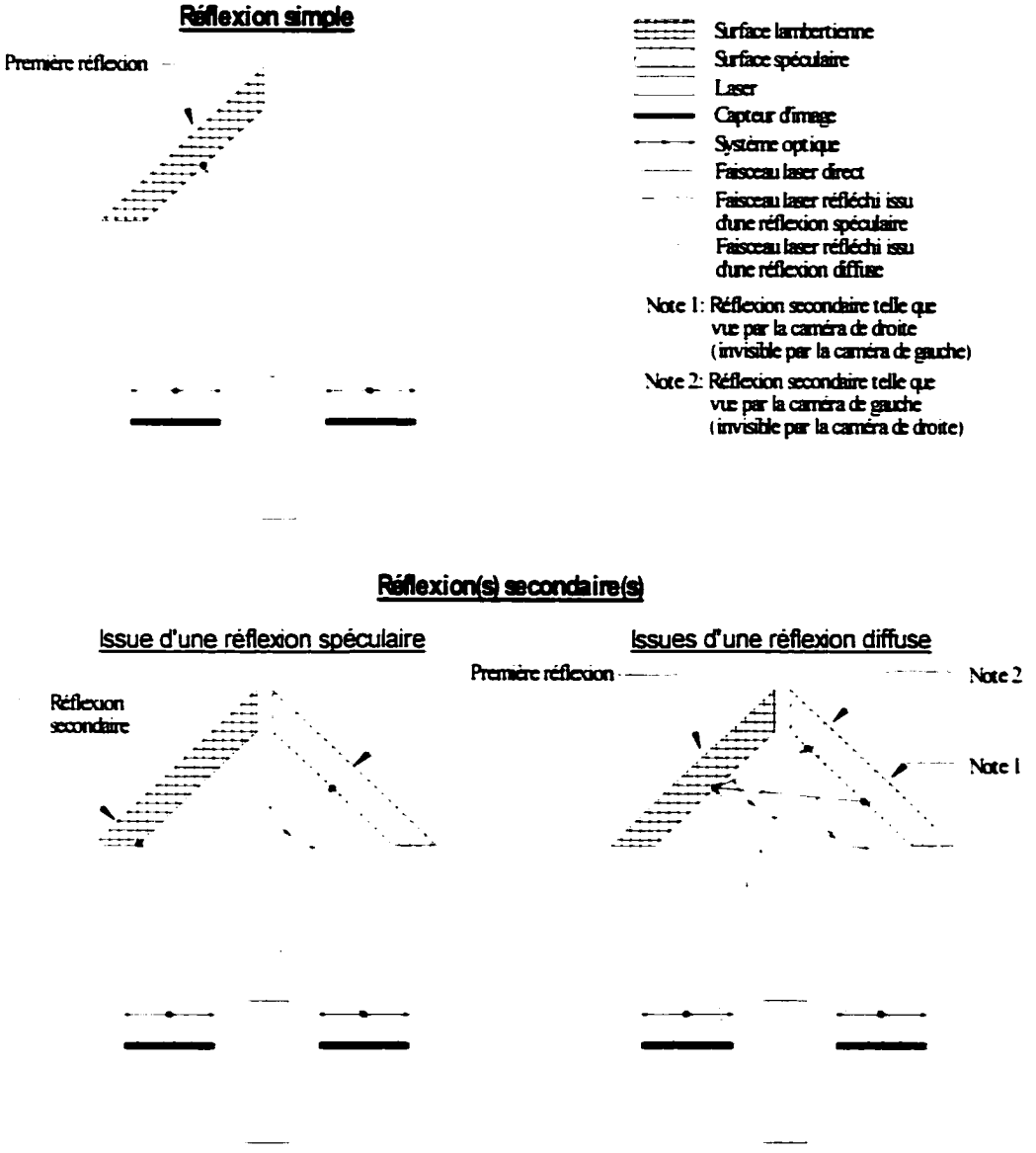


Figure 10 : Problématique reliée à la réflexion des points laser

Une courte réflexion sur le sujet permet de réaliser qu'il n'existe pas de solution simple à ce problème. Puisque le phénomène est difficile à identifier lorsqu'il se produit, il est d'autant plus difficile à solutionner. Une solution envisageable pouvant être implantée

consiste à faire une analyse contextuelle des données. Par exemple, appliquer un filtre de densité ou de dispersion spatiale sur le nuage de points pourrait évaluer la probabilité pour un point d'appartenir à une première ou à une $n^{ième}$ réflexion. Le développement d'un tel algorithme est complexe et ne peut garantir un filtrage optimal. Ce mémoire ne présente pas de solution à ce problème qui deviendra donc la grande lacune du système présenté.

D'un autre côté, malgré la puissance des sources de faisceaux laser, ce phénomène ne peut se produire que dans des situations très spécifiques. Il faut absolument qu'une surface hautement spéculaire comme un miroir ou une pièce métallique polie semblable au chrome soit présente sur la scène et qu'elle réfléchisse vers la caméra des points laser présents sur la scène. Sans ce type de situation, ce phénomène ne peut se produire car les réflexions diffuses sont trop faibles en intensité pour imager un point clair et défini sur l'image.

1.3.6 Problèmes de sécurité dus à la lumière laser

Lors du développement d'un outil aussi complexe qu'un robot mobile, le facteur sécurité prend des proportions importantes. Outre les parties mécaniques en mouvement et le déplacement du robot qui constituent la source principale de danger potentiel, le laser utilisé dans le système de vision est lui aussi dangereux. Dépendamment de la puissance du laser, il est possible d'endommager un oeil de façon sévère et permanente par un faisceau ou par sa réflexion entrant directement dans l'oeil.

On réalise rapidement que la puissance du laser correspond à une double contrainte contradictoire : un laser puissant pour contrer la lumière ambiante et un laser de faible intensité pour la sécurité des gens se trouvant près du robot. La section 3.1 présente le processus de réflexion permettant de sélectionner le laser utilisé pour le développement du prototype du capteur.

CHAPITRE 2

CONCEPTS THÉORIQUES DE LA VISION STÉRÉOSCOPIQUE

Étant donné que le capteur développé est basé sur le principe de la vision stéréoscopique, il est pertinent de présenter un bref rappel des notions théoriques du sujet. Il existe un très grand nombre d'ouvrages qui traitent de ce sujet et c'est pourquoi cette section ne fait qu'un bref survol des concepts importants. Le lecteur intéressé peut consulter Ayache (1989), Horaud et Monga (1993) et Trucco et Verri (1998).

La vision stéréoscopique consiste à utiliser *n* images de la même scène dont le point de vue diffère pour identifier la troisième dimension. Le principe consiste simplement à jumeler une observation se trouvant dans au moins deux images et correspondant à un seul élément du monde réel. À l'aide des coordonnées images, on calcule la position spatiale. Un système de vision stéréoscopique passe principalement par cinq étapes importantes :

1. *Le calibrage.* Le calibrage du système permet de connaître les relations existant entre les coordonnées spatiales en trois dimensions et les coordonnées images en deux dimensions. Cette technique est réalisée par une modélisation du système optique et c'est par échantillonnage que les paramètres du modèle sont évalués.
2. *L'acquisition des images.* Le processus débute par la projection de la scène réelle 3D sur un plan 2D par un système optique. Ensuite, l'image est captée et transférée à l'ordinateur qui traite le tout de façon numérique.

3. *L'extraction des observations présentes sur les images.* Cette étape permet de trouver sur les images les constituants qui décrivent la scène. Dans le cas présent, ce sont les images des points laser qui sont les observations recherchées.

4. *La rectification et l'appariement.* L'appariement consiste à identifier la même caractéristique de la scène sur chacune des *n* images. Une étape importante pour réaliser ce jumelage consiste à modifier les images en fonction de la géométrie particulière d'un système de vision stéréoscopique. C'est ce qu'on appelle la rectification.

5. *La reconstruction.* Elle consiste à évaluer la coordonnée spatiale 3D à partir des coordonnées 2D des observations présentes sur les *n* images.

2.1 Le modèle de la caméra

La géométrie relativement simple du système optique permet de trouver la relation qui existe entre une coordonnée spatiale et la coordonnée du point image résultant. Il existe dans la littérature plusieurs modèles de caméra très semblables. Le modèle présenté ici correspond à un sténopé non-inverseur et divise la liste des paramètres en deux groupes : les paramètres intrinsèques et les paramètres extrinsèques.

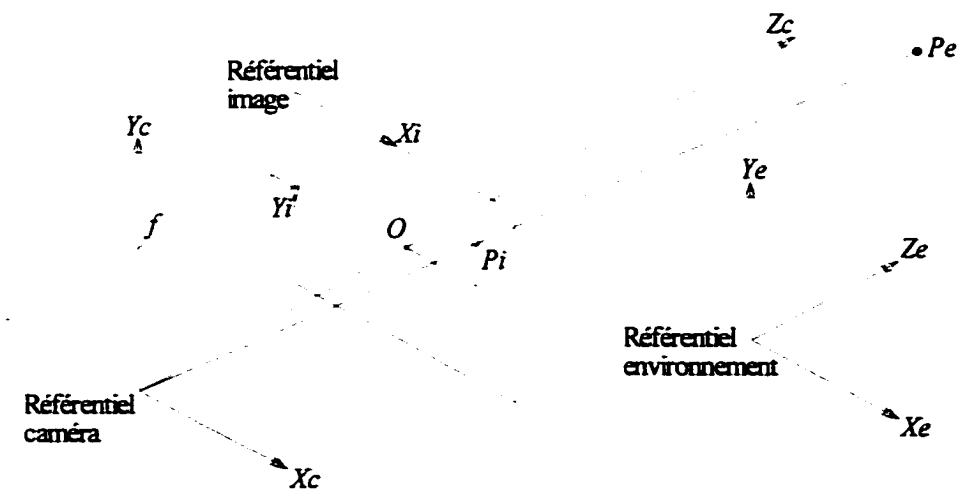


Figure 11 : Modèle de la caméra

où X_e, Y_e et Z_e représentent le référentiel de l'environnement

X_c, Y_c et Z_c représentent le référentiel de la caméra

X_i et Y_i représentent le référentiel image

f est la distance focale

$O_{x,y}$ est le centre de l'image (où l'axe optique croise le CCD)

P_e est un point dans l'environnement

P_i est l'image du point P_e

2.1.1 Les paramètres intrinsèques

Ces paramètres permettent d'établir la relation entre un point du référentiel caméra par rapport au référentiel image. Ces paramètres sont la distance focale (f), le centre optique ($O_{x,y}$), les facteurs d'échelle représentant la dimension réelle des pixels ($s_{x,y}$), l'angle réel entre les axes x et y du référentiel image (θ) et finalement la distorsion radiale (r) causée par les aberrations géométriques de certaines lentilles.

La qualité actuelle des caméras CCD fait en sorte qu'on suppose le paramètre θ égal à 90°. Ainsi il est possible de négliger ce paramètre dans le calibrage. Il est aussi possible de négliger r sous certaines conditions. La distorsion radiale est un phénomène déformant les images par la géométrie du système optique. Ce phénomène est plus intense en périphérie des images et plus important avec des objectifs à moyen et surtout à grand angle. Si le système optique est constitué de lentilles à petit angle, il est donc possible d'omettre ce paramètre.

En négligeant ces deux paramètres, les équations suivantes décrivent la relation entre le référentiel caméra et le référentiel image :

$$x_i = o_x - \frac{f \cdot x_c}{s_x \cdot z_c}; \quad y_i = o_y - \frac{f \cdot y_c}{s_y \cdot z_c} \tag{2.1}$$

Malheureusement, tous les paramètres précédents ne sont pas mutuellement indépendants. En posant $f_x = \frac{f}{s_x}$ et $f_y = \frac{f}{s_y}$, les paramètres deviennent indépendants. En manipulant ces équations, la forme matricielle suivante est obtenue :

$$M_{int} = \begin{bmatrix} -f_x & 0 & o_x \\ 0 & -f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

2.1.2 Les paramètres extrinsèques

Les paramètres extrinsèques permettent le passage des coordonnées de l'environnement aux coordonnées de la caméra. Cette transformation est beaucoup plus simple puisqu'il suffit d'appliquer une rotation et une translation dans l'espace. Sous forme matricielle, on a :

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{2.3}$$

où : t_x, t_y et t_z représentent les éléments du vecteur de translation

r_{ab} représentent les éléments de la matrice de rotation (avec rotation de l'axe a par rapport à l'axe b)

2.1.3 La matrice de projection perspective

Connaissant les paramètres intrinsèques et extrinsèques, il est maintenant possible de calculer la position d'un point se trouvant dans l'environnement. Ainsi on peut calculer la matrice de projection M qui correspond à $M_{int} \times M_{ext}$.

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = M_{int} \cdot M_{ext} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = M \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -f_x r_{11} + o_x r_{31} & -f_x r_{12} + o_x r_{32} & -f_x r_{13} + o_x r_{33} & -f_x t_x + o_x t_z \\ -f_y r_{21} + o_y r_{31} & -f_y r_{22} + o_y r_{32} & -f_y r_{23} + o_y r_{33} & -f_y t_y + o_y t_z \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \cdot \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}$$

(2.4)

L'usage des coordonnées homogènes donne comme solution : $x_i = \frac{a}{c}$ et $y_i = \frac{b}{c}$.

2.2 Calibrage

Il est essentiel d'évaluer la matrice de projection perspective M car c'est elle qui permet de déterminer la relation existant entre un point de l'espace et un point image.

La façon d'évaluer M se fait par échantillonnage. Il suffit de connaître la position de n points de l'espace et de leur image pour résoudre le système d'équation 2.4.

La matrice de projection perspective contient douze paramètres. Or Faugeras et Toscani (1986) proposent de poser t_z non nul et égal à 1 puisque tous les paramètres sont définis à un facteur d'échelle près. Ainsi on a 11 inconnus à déterminer, c'est-à-dire m_{11}, m_{12}, m_{13}, m_{14}, m_{21}, m_{22}, m_{23}, m_{24}, m_{31}, m_{32} et m_{33}. Sachant qu'il y a deux équations pour chaque point échantillonné, il suffit de six points pour résoudre M. Néanmoins, il est fortement recommandé, voire essentiel, d'utiliser le maximum de points et de résoudre le système par des techniques comme les moindres carrés ou le filtre de Kalman tel que présenté par Ayache (1989).

On pose le système d'équation linéaire suivant :

$$a \cdot M = b \tag{2.5}$$

où :

$$a = \begin{bmatrix} x_{e_1} & y_{e_1} & z_{e_1} & 1 & 0 & 0 & 0 & 0 & -x_{e_1}x_{i_1} & -y_{e_1}x_{i_1} & -z_{e_1}x_{i_1} \\ 0 & 0 & 0 & 0 & x_{e_1} & y_{e_1} & z_{e_1} & 1 & -x_{e_1}y_{i_1} & -y_{e_1}y_{i_1} & -z_{e_1}y_{i_1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{e_i} & y_{e_i} & z_{e_i} & 1 & 0 & 0 & 0 & 0 & -x_{e_i}x_{i_i} & -y_{e_i}x_{i_i} & -z_{e_i}x_{i_i} \\ 0 & 0 & 0 & 0 & x_{e_i} & y_{e_i} & z_{e_i} & 1 & -x_{e_i}y_{i_i} & -y_{e_i}y_{i_i} & -z_{e_i}y_{i_i} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{e_n} & y_{e_n} & z_{e_n} & 1 & 0 & 0 & 0 & 0 & -x_{e_n}x_{i_n} & -y_{e_n}x_{i_n} & -z_{e_n}x_{i_n} \\ 0 & 0 & 0 & 0 & x_{e_n} & y_{e_n} & z_{e_n} & 1 & -x_{e_n}y_{i_n} & -y_{e_n}y_{i_n} & -z_{e_n}y_{i_n} \end{bmatrix} \quad b = \begin{bmatrix} x_{i_1} \\ y_{i_1} \\ \cdots \\ x_{i_i} \\ y_{i_i} \\ \cdots \\ x_{i_n} \\ y_{i_n} \end{bmatrix}$$

La méthode des moindres carrés se résume dans ce cas par la pseudo-inverse de a appliquée à b.

$$M = (a^t a)^{-1} a^t b \tag{2.6}$$

Il n'est pas nécessaire de connaître explicitement la valeur de chacun des paramètres intrinsèques et extrinsèques car pour résoudre le système de vision stéréoscopique, il suffit d'évaluer les matrices de projection perspective. Malgré tout, il est parfois utile et même essentiel de connaître explicitement la valeur de chacun de ces paramètres. Plusieurs ouvrages traitent de ce sujet. Le lecteur intéressé peut se référer à Tsai (1983), à Horaud et Monga (1993) et à Trucco et Verri (1998).

2.3 Géométrie épipolaire et rectification des images

Tel qu'illustré sur la figure ci-dessous, on suppose un point sur l'image de gauche (I_g) à partir duquel on trace une droite dans l'espace. Cette droite correspond aux positions potentielles du point réel dans l'espace. Peu importe la position du point P sur cette

droite. le point image I_g est le même. Pour déterminer cette droite, il suffit de trouver celle qui passe par le point image I_g et le centre optique de la caméra C_g.

Figure 12 : Projection de la droite 3D provenant d'un point image

Le centre optique d'une caméra est très facile à trouver avec la matrice de projection perspective M. Il suffit de résoudre le système suivant :

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = M \cdot \begin{bmatrix} C_x \\ C_y \\ C_z \\ 1 \end{bmatrix} \tag{2.7}$$

où : C est la coordonnée du centre optique de la caméra définie selon le référentiel de l'environnement

M est la matrice de projection perspective

Pour chaque position potentielle de P sur la droite passant par I_g et C_g , on trace un point sur l'image de droite. De cette série de points naît la droite épipolaire qui représente les positions potentielles du point image droite correspondant au point image gauche.

Figure 13 : Construction de la droite épipolaire

C'est avec les droites épipolaires et les centres optiques des caméras qu'il est possible de déterminer les épipôles et les points de convergence des différentes droites épipolaires. La prochaine figure montre la géométrie épipolaire dans son entier. On y retrouve les centres optiques (C_d et C_g) ainsi que les épipôles (E_d et E_g).

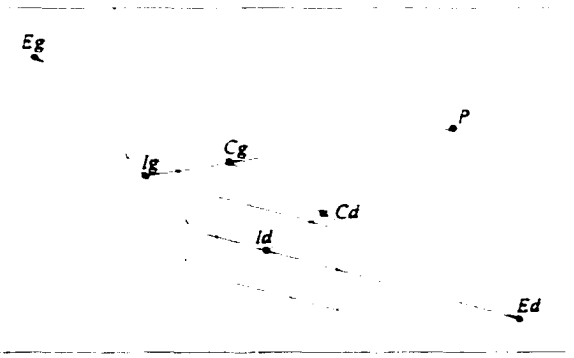


Figure 14 : Géométrie épipolaire

La rectification des images consiste à appliquer une transformation linéaire en coordonnées projectives aux images de façon à ramener les épipôles à l'infini. Cette géométrie est entièrement dépendante du système optique et de la configuration spatiale relative des caméras entre elles.

En repoussant les épipôles à l'infini, de nouvelles images I_d' et I_g' sont créées. Ces images, déformées de la réalité, possèdent un avantage : un point de l'environnement se situe exactement aux mêmes ordonnées sur les deux images.

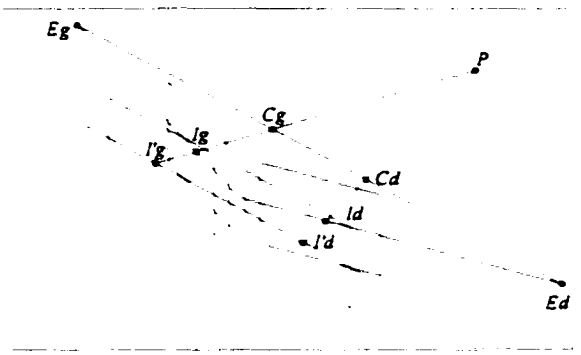


Figure 15 : Rectification

Plusieurs techniques peuvent être employées pour rectifier une image. Certaines utilisent directement les paramètres intrinsèques et extrinsèques mais la technique utilisée ici est basée uniquement sur la géométrie épipolaire. Cette opération transforme les coordonnées (x, y) de chaque pixel en nouvelles coordonnées (x', y'). Ayache (1989) démontre que cette transformation peut s'effectuer de la façon suivante :

$$\begin{bmatrix} \frac{x_g'}{s} \\ \frac{y_g'}{s} \\ s \end{bmatrix} = \begin{bmatrix} ((C_g \wedge C_d) \wedge C_g)^t \\ (C_g \wedge C_d)^t \\ ((C_g - C_d) \wedge (C_g \wedge C_d))^t \end{bmatrix} \cdot \left[x_g (m_{2g} \wedge m_{3g}) + y_g (m_{3g} \wedge m_{1g}) + (m_{1g} \wedge m_{2g}) \right] \tag{2.8}$$

$$\begin{bmatrix} \frac{x_d'}{s} \\ \frac{y_d'}{s} \\ s \end{bmatrix} = \begin{bmatrix} ((C_g \wedge C_d) \wedge C_d)^t \\ (C_g \wedge C_d)^t \\ ((C_g - C_d) \wedge (C_g \wedge C_d))^t \end{bmatrix} \cdot [x_d(m_{2d} \wedge m_{3d}) + y_d(m_{3d} \wedge m_{1d}) + (m_{1d} \wedge m_{2d})] \tag{2.9}$$

où d est l'indice correspondant aux paramètres de la caméra de droite

g est l'indice correspondant aux paramètres de la caméra de gauche

C_d et C_g correspondent aux centres optiques des caméras de droite et de gauche définis dans le référentiel de l'environnement

m_{1g} correspond au vecteur $[m_{11}\ m_{12}\ m_{13}]$ de M de la caméra de gauche

m_{1d} correspond au vecteur $[m_{11}\ m_{12}\ m_{13}]$ de M de la caméra de droite

$\wedge$ est l'opérateur du produit vectoriel

En pratique, il n'est pas nécessaire de rectifier complètement une image. Il suffit de rectifier les zones pertinentes ou simplement les coordonnées des observations importantes des images.

2.4 Appariement

L'appariement est certainement la tâche la plus délicate et la plus complexe du processus de vision stéréoscopique. En effet, il ne suffit pas d'extraire les observations importantes des deux images pour reconstruire la troisième dimension. Il importe de jumeler les observations entre elles. Il est essentiel de s'assurer que les paires de points images (gauche et droite) correspondent à la même région de la scène, sinon la reconstruction 3D ne peut représenter fidèlement la scène. La figure suivante illustre la complexité associée à cette tâche. En prenant le point image 1 de la caméra de gauche et les trois points images présents sur la caméra de droite, un maximum de trois points réels peuvent correspondre à une telle combinaison. Lequel des points images de la caméra de droite correspond réellement au point image 1 de la caméra de gauche? Le

problème se corse d'avantage lorsque plus de points sont présents sur la caméra de gauche. En fait, le nombre d'appariements exclusifs possibles est égal au produit du nombre d'observations présentes sur chacune des images.

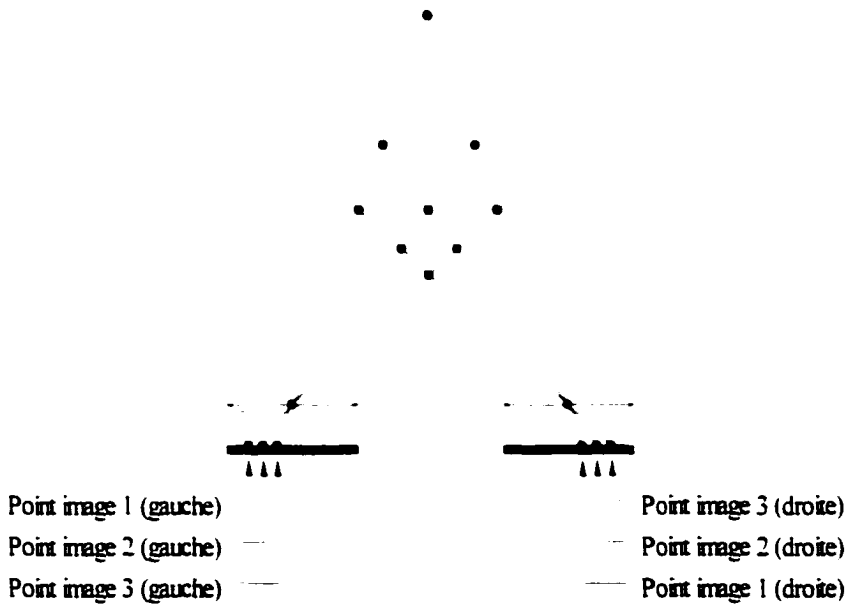


Figure 16 : Problématique associée au problème d'appariement

Pour limiter le nombre d'appariements possibles, il suffit de restreindre les associations par différentes contraintes. Une revue de la littérature montre qu'il existe une quantité importante de contraintes permettant de limiter les erreurs d'appariements. Les sections suivantes passent en revue l'ensemble des contraintes existantes. On approfondit les plus importantes et les plus pertinentes pour le développement du capteur.

2.4.1 Contrainte épipolaire

La section précédente présente les concepts associés à la géométrie épipolaire. La contrainte épipolaire est la seule contrainte géométrique intrinsèque du capteur stéréoscopique. À l'aide de cette propriété géométrique, il est possible de rectifier les

images et de retrouver sur les mêmes ordonnées des deux images les observations qui peuvent s'apparier.

La géométrique épipolaire présente deux caractéristiques intéressantes. Premièrement, la différence des ordonnées de deux points laser sur les deux images indique le niveau de probabilité selon lequel ces points correspondent entre eux. Deuxièmement, la différence des abscisses de ces points, appelée disparité, est inversement proportionnelle à la distance du point P observé.

Ainsi cette contrainte géométrique est très significative et permet de limiter de façon importante les appariements possibles. La section 4.3.5 montre comment la méthode implantée pour l'algorithme d'appariement tient compte de cette contrainte.

2.4.2 Intervalle de disparité autorisé

Tel que constaté précédemment, la géométrie épipolaire permet de restreindre considérablement le nombre d'appariements possibles. Il est possible de profiter encore des caractéristiques géométriques du système optique pour limiter davantage ce nombre d'appariements.

On peut définir la disparité par :

$$\delta = x_g^r - x_d^r \tag{2.10}$$

où x_g^r *est l'abscisse rectifiée de l'image de gauche*
x_d^r *est l'abscisse rectifiée de l'image de droite*

La figure suivante illustre les positions limites d'un point image I_d correspondant aux positions limites d'un point P sur la droite passant par I_g et C_g. Le point image situé en k_{min} correspond à l'image du point P si se dernier se trouve à une distance infinie. Le point image situé en k_{max} correspond à l'image du point P si ce dernier se trouve

directement sur le centre optique de la caméra de gauche. On remarque que k_{max} est, par définition, l'épipôle. La droite tracée et limitée par k_{min} et k_{max} définit l'intervalle de disparité autorisé. Ainsi un point image se trouvant au-delà des limites de cette droite n'a aucun sens physique et ne peut faire partie des appariements possibles.

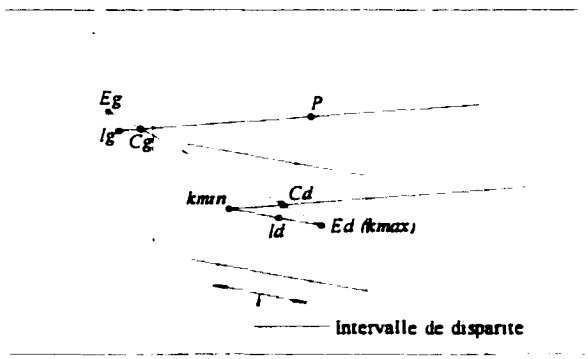


Figure 17 : Limite de l'intervalle de disparité

où C_g et C_d sont les centres optiques des caméras de gauche et de droite
E_g et E_d sont les épipôles des caméras de gauche et de droite
P est le point dans l'espace et I_g et I_d sont les points image de P
k_{min} et k_{max} sont les limites de l'intervalle de disparité autorisé

Pour calculer k_{min} et k_{max} , il faut d'abord calculer l'équation de la droite de l'espace passant par C_g et par P. Puisque P n'est pas connu, on calcule la droite à partir de I_g et de M. De l'équation 2.4, on peut écrire les équations définissant les plans dont l'intersection est la droite P :

$$\left(M_{11}^{g} - x^{g} M_{13}^{g}\right)P_x + \left(M_{21}^{g} - x^{g} M_{23}^{g}\right)P_y + \left(M_{31}^{g} - x^{g} M_{33}^{g}\right)P_z + M_{41}^{g} - x^{g} M_{43}^{g} = 0$$
$$\left(M_{12}^{g} - y^{g} M_{13}^{g}\right)P_x + \left(M_{22}^{g} - y^{g} M_{23}^{g}\right)P_y + \left(M_{32}^{g} - y^{g} M_{33}^{g}\right)P_z + M_{42}^{g} - y^{g} M_{43}^{g} = 0$$

En posant $m_i = \begin{bmatrix} m_{1i} & m_{2i} & m_{3i} \end{bmatrix}$, on peut réécrire les équations :

$$\begin{aligned}\left(m_1^g - x^g m_3^g\right)P + M_{41}^g - x^g M_{43}^g = 0 \\ \left(m_2^g - y^g m_3^g\right)P + M_{42}^g - y^g M_{43}^g = 0\end{aligned} \tag{2.11}$$

où : M^g est la matrice de projection perspective de la caméra de gauche
P est le point dans l'espace
(x^g, y^g) est la coordonnée de I_g sur l'image de gauche

On calcule le vecteur directeur n de la droite passant par P par le produit vectoriel des vecteurs normaux correspondants aux plans. On obtient ainsi :

$$n = \left(m_1^d - x^d m_3^d\right) \wedge \left(m_2^d - y^d m_3^d\right) \tag{2.12}$$

L'équation paramétrique de la droite est $P = C_g + \lambda \cdot n$. On calcule le point I^d sur l'image de droite avec la matrice de projection perspective de la caméra de droite (M^d) par :

$$I^d = M^d \cdot \begin{bmatrix} C_g + \lambda \cdot n \\ 1 \end{bmatrix} \tag{2.13}$$

On calcule k_{min} en posant $\lambda \to \infty$ et k_{max} en posant $\lambda \to 0$. On remarque encore que la définition de k_{max} correspond à l'épipôle (E_d).

$$k_{\max} = M^d \cdot \begin{bmatrix} C_g \\ 1 \end{bmatrix} \tag{2.14}$$

Pour évaluer k_{min}, on développe l'équation 2.13 qui donne :

$$k_{\min x} = \frac{M_{11}^d\left(C_x^g + \lambda \cdot n_x\right) + M_{12}^d\left(C_y^g + \lambda \cdot n_y\right) + M_{13}^d\left(C_z^g + \lambda \cdot n_z\right) + M_{14}^d}{M_{31}^d\left(C_x^g + \lambda \cdot n_x\right) + M_{32}^d\left(C_y^g + \lambda \cdot n_y\right) + M_{33}^d\left(C_z^g + \lambda \cdot n_z\right) + M_{34}^d}$$

$$k_{\min y} = \frac{M_{21}^d\left(C_x^g + \lambda \cdot n_x\right) + M_{22}^d\left(C_y^g + \lambda \cdot n_y\right) + M_{23}^d\left(C_z^g + \lambda \cdot n_z\right) + M_{24}^d}{M_{31}^d\left(C_x^g + \lambda \cdot n_x\right) + M_{32}^d\left(C_y^g + \lambda \cdot n_y\right) + M_{33}^d\left(C_z^g + \lambda \cdot n_z\right) + M_{34}^d}$$

$$k_{\min x} = \frac{\lambda \cdot \left(M_{11}^d n_x + M_{12}^d n_y + M_{13}^d n_z\right) + M_{11}^d C_x^g + M_{12}^d C_y^g + M_{13}^d C_z^g + M_{14}^d}{\lambda \cdot \left(M_{31}^d n_x + M_{32}^d n_y + M_{33}^d n_z\right) + M_{31}^d C_x^g + M_{32}^d C_y^g + M_{33}^d C_z^g + M_{34}^d}$$

$$k_{\min y} = \frac{\lambda \cdot \left(M_{21}^d n_x + M_{22}^d n_y + M_{23}^d n_z\right) + M_{21}^d C_x^g + M_{22}^d C_y^g + M_{23}^d C_z^g + M_{24}^d}{\lambda \cdot \left(M_{31}^d n_x + M_{32}^d n_y + M_{33}^d n_z\right) + M_{31}^d C_x^g + M_{32}^d C_y^g + M_{33}^d C_z^g + M_{34}^d}$$

En calculant la limite à l'infini, on a :

$$k_{\min x} = \lim_{\lambda \to \infty} \frac{\frac{\lambda \cdot \left(M_{11}^d n_x + M_{12}^d n_y + M_{13}^d n_z\right)}{\lambda} + \frac{M_{11}^d C_x^g + M_{12}^d C_y^g + M_{13}^d C_z^g + M_{14}^d}{\lambda}}{\frac{\lambda \cdot \left(M_{31}^d n_x + M_{32}^d n_y + M_{33}^d n_z\right)}{\lambda} + \frac{M_{31}^d C_x^g + M_{32}^d C_y^g + M_{33}^d C_z^g + M_{34}^d}{\lambda}}$$

$$k_{\min y} = \lim_{\lambda \to \infty} \frac{\frac{\lambda \cdot \left(M_{21}^d n_x + M_{22}^d n_y + M_{23}^d n_z\right)}{\lambda} + \frac{M_{21}^d C_x^g + M_{22}^d C_y^g + M_{23}^d C_z^g + M_{24}^d}{\lambda}}{\frac{\lambda \cdot \left(M_{31}^d n_x + M_{32}^d n_y + M_{33}^d n_z\right)}{\lambda} + \frac{M_{31}^d C_x^g + M_{32}^d C_y^g + M_{33}^d C_z^g + M_{34}^d}{\lambda}}$$

Et finalement, on obtient ces équations :

$$k_{\min x} = \frac{M_{11}^d n_x + M_{12}^d n_y + M_{13}^d n_z}{M_{31}^d n_x + M_{32}^d n_y + M_{33}^d n_z} = \frac{m_1^d n}{m_3^d n}$$

$$k_{\min x} = \frac{M_{21}^d n_x + M_{22}^d n_y + M_{23}^d n_z}{M_{31}^d n_x + M_{32}^d n_y + M_{33}^d n_z} = \frac{m_2^d n}{m_3^d n} \qquad (2.15)$$

2.4.3 Contrainte d'ordre

La contrainte d'ordre considère l'ordre d'apparition des points dans les images. Cette contrainte est basée sur l'hypothèse suivante : les ensemble Q_d et Q_g possèdent le même ordre d'apparition des éléments de la scène. Les ensembles de points $Q_g = \{p_1^g, p_2^g, p_3^g, \ldots\}$ et $Q_d = \{p_1^d, p_2^d, p_3^d, \ldots\}$ correspondent aux images de gauche et de droite. Chaque point de ces ensembles se trouve sur les droites épipolaires conjuguées.

Cette contrainte n'est malheureusement pas absolue et se base sur de nombreuses hypothèses. Par exemple, il faut supposer que les objets perçus ne sont pas de nature transparente et que la partie avant cache bien la partie arrière. Car dans ces cas, la contrainte d'ordre n'est plus valide.

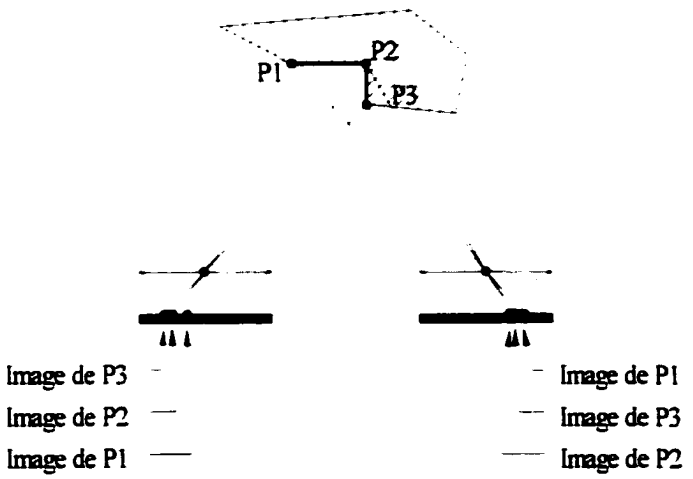


Figure 18 : Contrainte d'ordre (problématique reliée à la transparence et aux surfaces fortement inclinées)

La figure précédente illustre bien cette problématique. En supposant que l'objet représenté par la zone hachurée soit de nature opaque, on retrouve sur l'image de gauche trois points images tandis qu'on retrouve seulement l'image des points P_1 et P_3 sur l'image de droite. Dans ce cas, la contrainte d'ordre est respectée car l'ordre

d'apparition dans les ensembles est respecté ($Q_g = \{P_3^g, P_2^g, P_1^g\}$ et $Q_d = \{P_3^d, P_1^d\}$). D'un autre côté, si l'objet représenté par la zone hachurée est de nature transparente, la contrainte d'ordre n'est pas respectée car l'ordre d'apparition dans les ensembles n'est pas respecté ($Q_g = \{P_3^g, P_2^g, P_1^g\}$ et $Q_d = \{ P_2^d, P_3^d, P_1^d\}$).

De plus, il existe de nombreuses formes géométriques qui, même étant opaques, rendent la contrainte d'ordre peu efficace. Étant donné que la disparité est inversement proportionnelle à la distance, les objets éloignés ont pratiquement les mêmes abscisses sur les deux images donc un indice de disparité très faible ($\delta \rightarrow 0$). D'un autre côté, les objets près des caméras ont un indice de disparité beaucoup plus grand. La figure suivante montre une paire d'image stéréoscopique qui ne respecte pas la contrainte d'ordre. Les groupes $Q_g = \{P_3^g, P_1^g, P_4^g, P_2^g\}$ et $Q_d = \{P_1^d, P_3^d, P_2^d, P_4^d\}$ diffèrent simplement à cause de l'effet de perspective appliqué à une forme géométrique ayant plusieurs concavités. Naturellement, cet exemple précis peut être simplement solutionné en considérant les droites épipolaires.

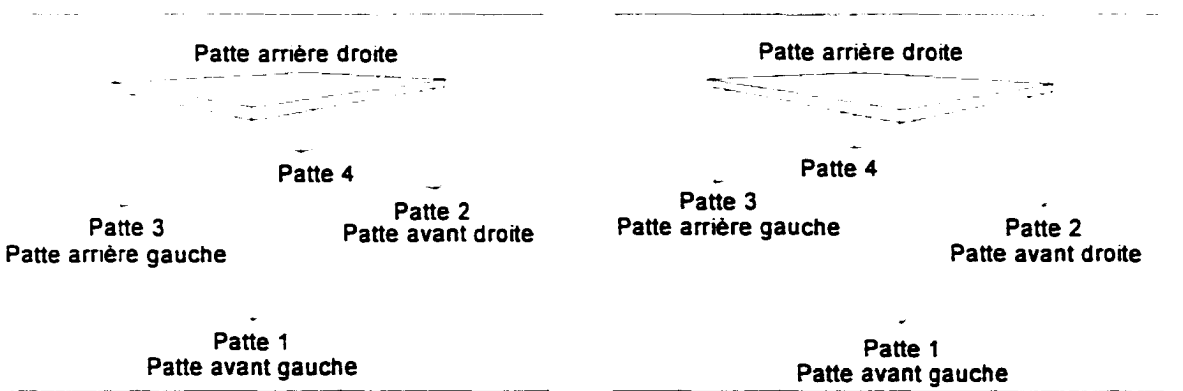


Figure 19 : Contrainte d'ordre (problématique reliée à la perspective)

Dans le cas d'une application générique, cette contrainte est difficilement justifiable puisque plusieurs situations ne peuvent respecter les hypothèses de base. Ainsi on ne considère pas la contrainte d'ordre pour solutionner le problème d'appariement.

2.4.4 Limite du gradient de disparité

La limite du gradient de disparité est une extension du concept associé à la contrainte d'ordre. La contrainte d'ordre est aussi associée à l'inclinaison d'une surface observée par rapport au plan des deux images (voir la figure 18). Il est possible d'exprimer cette contrainte d'une autre façon en tenant compte de la variation maximum de profondeur qui permet de respecter la contrainte d'ordre.

Puisque cette contrainte est basée sur la contrainte d'ordre, elle n'est pas utilisée pour l'implémentation finale. Le lecteur intéressé peut se référer à Ayache (1989) et Horaud et Monga (1993).

2.4.5 Contrainte d'unicité

La contrainte d'unicité est basée sur l'affirmation suivante : il existe un et un seul point réel de l'environnement correspondant à une paire de points image. Ainsi un point P de l'espace est constitué d'un seul et unique point sur chaque image. De plus, en respectant la contrainte d'unicité, chacun des points sur les images ne peut décrire plus d'un point dans l'espace. Encore une fois, cette contrainte est basée sur l'hypothèse selon laquelle il n'existe pas de variation de surface trop brusque. Si on considère que la surface décrite à la figure 18 est opaque, on peut voir que P_2 ne peut être imagé sur l'image de droite.

L'affirmation sur laquelle repose la contrainte d'unicité est directement reliée à la contrainte d'ordre. En fait, la contrainte d'unicité est le cas limite du gradient de disparité. Si on se positionne à la frontière de la limite permettant de respecter le gradient de disparité, on voit que deux points de l'espace peuvent être imagés à deux endroits sur une image et à un seul endroit sur l'autre image. La figure suivante illustre ce cas limite.

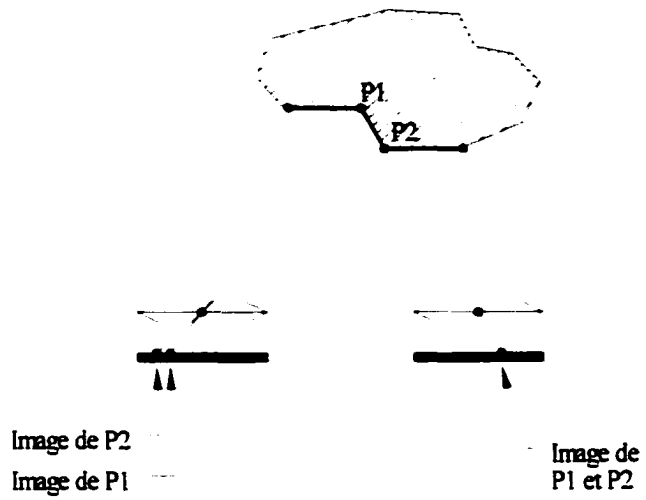


Figure 20 : Contrainte d'unicité

Malgré le désir de résoudre une application générique pouvant solutionner la majorité des cas possibles, l'utilisation de la contrainte d'unicité est préférable sinon le problème d'appariement devient beaucoup plus complexe. De plus, les occasions où ces cas se présentent sont plutôt rares. Pour toutes ces raisons, cette contrainte est utilisée dans les algorithmes décrits ultérieurement.

2.4.6 Contrainte d'orientation

La contrainte d'orientation est basée sur l'orientation principale des observations analysées se trouvant sur l'image. Ainsi le vecteur de l'orientation principale permet de trouver l'équivalent le plus près dans la seconde image lorsque les deux images sont rectifiées.

Cette contrainte n'est pas très utile pour l'application qui nous occupe car la structure même des observations ne possède pas d'orientation principale. Les seules observations considérées sur les images sont des petits points circulaires. Ainsi cette contrainte ne peut être utilisée.

2.4.7 Contrainte de continuité figurale

La contrainte de continuité figurale est basée sur la continuité des points se trouvant sur un contour. Les points se trouvant sur un même contour sur l'image de gauche doivent trouver leur correspondant sur un même contour sur l'image de droite. Encore une fois, cette contrainte ne peut être utilisée puisque la nature des observations ne permet pas de faire partie des éléments d'un contour.

2.4.8 Vision stéréoscopique trinoculaire

Les contraintes présentées précédemment sont toutes des contraintes reliées à la vision stéréoscopique binoculaire. L'utilisation d'une troisième caméra (où même d'une $n^{ième}$ caméras, où $n > 2$) permet de résoudre complètement le problème d'appariement.

Avec un tel type de capteur, la première étape consiste à poser une hypothèse d'appariement entre deux points basés minimalement sur la contrainte épipolaire : ces points se retrouvent le long des droites épipolaires des caméras associées. Ensuite, il suffit de valider cette hypothèse en vérifiant s'il existe un point homologue sur la troisième image à l'intersection des droites épipolaires conjuguées. La figure suivante illustre la géométrie épipolaire pour un système de vision stéréoscopique trinoculaire.

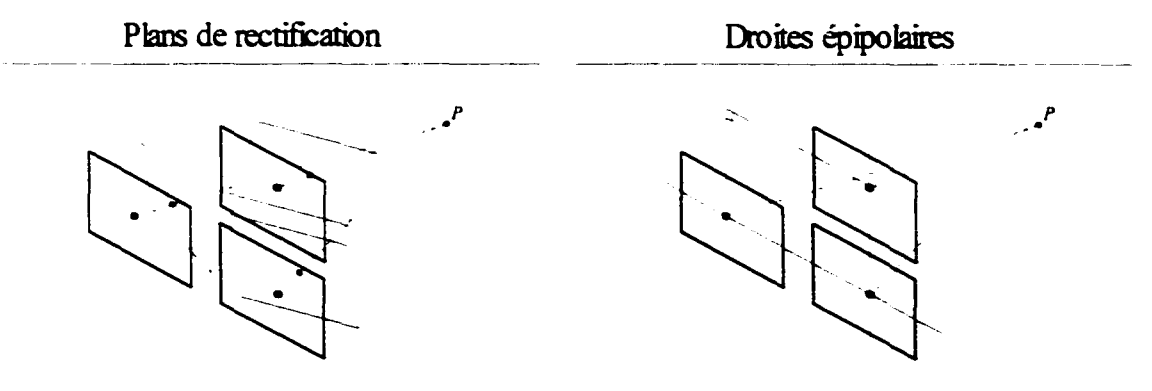


Figure 21 : Géométrie épipolaire pour un système stéréoscopique trinoculaire

2.4.9 Contrainte de corrélation de l'intensité entre deux régions

Cette contrainte est la seule qui compare la ressemblance entre les régions d'intérêt des deux images. On évalue la ressemblance entre ces deux régions par un simple calcul de corrélation sur l'éclairement lumineux des images.

$$Corr\left(I^g, I^d\right) = \sum_{i=-M}^{M} \sum_{j=-N}^{N} \left(I^g\left(x^g - i, y^g - j\right) - I^d\left(x^d - i, y^d - j\right)\right)^2 \qquad (2.16)$$

où : $Corr(I^g, I^d)$ est l'indice de corrélation calculé sur les images de gauche (I^g) et de droite (I^d)

x^g et y^g indiquent le centre de la région d'intérêt se trouvant sur l'image de gauche

x^d et y^d indiquent le centre de la région d'intérêt se trouvant sur l'image de droite

$(2M + 1) \times (2N + 1)$ est la taille de la région analysée

Malgré tout, l'efficacité de cette contrainte est faible pour des points et forte pour des formes plus complexes. Par contre, elle reste suffisamment performante pour aider à l'appariement.

2.5 Reconstruction 3D

La reconstruction tridimensionnelle est l'étape ultime du processus de vision stéréoscopique. Elle consiste à retrouver la position du point dans l'espace correspondant aux deux points images trouvés et appariés.

Il existe plusieurs méthodes simples pour retrouver les coordonnées spatiales du point. Deux sont présentées ici. La première, nommée reconstruction par le point milieu des droites directrices, est basée sur la géométrie du système. La seconde, nommée reconstruction par la technique des moindres carrés, est basée sur une méthode d'estimation permettant de minimiser l'erreur.

2.5.1 Reconstruction par le point milieu des droites directrices

Cette méthode de reconstruction très simple est basée sur la géométrie du système. Connaissant les deux points images, il suffit de calculer l'intersection des droites de projection perspective issues des points images. Par contre, il est fort probable que les erreurs de mesure et les arrondissements numériques fassent en sorte qu'il n'existe pas d'intersection. Pour résoudre ce problème, il suffit de prendre le point milieu du plus petit segment de droite touchant aux deux droites et perpendiculaire à celles-ci. Ce point milieu correspond au point *P*.

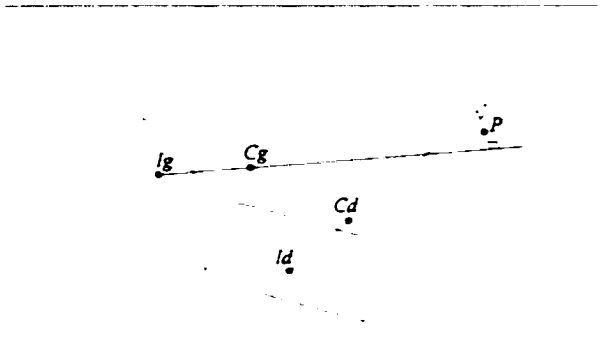


Figure 22 : Reconstruction 3D par le point milieu des droites directrices

Le lecteur intéressé peut se référer à Trucco et Verri (1998) qui montrent comment effectuer cette opération.

2.5.2 Reconstruction par la technique des moindres carrés

La deuxième méthode est basée sur la même démarche que celle utilisée lors du calibrage. De l'équation 2.4, on peut écrire ce nouveau système d'équation :

$$\begin{aligned}
\left(m_1^g - i_x^g m_3^g\right)^T P + m_{14}^g - i_x^g m_{34}^g &= 0 \\
\left(m_2^g - i_y^g m_3^g\right)^T P + m_{24}^g - i_y^g m_{34}^g &= 0 \\
\left(m_1^d - i_x^d m_3^d\right)^T P + m_{14}^d - i_x^d m_{34}^d &= 0 \\
\left(m_2^d - i_y^d m_3^d\right)^T P + m_{24}^d - i_y^d m_{34}^d &= 0
\end{aligned} \tag{2.17}$$

Ce système d'équation donne maintenant quatre équations et trois inconnus (le vecteur P qui est le point recherché [x y z]). La méthode des moindres carrés est utilisée pour résoudre le système.

$$aP = b \tag{2.18}$$

où : $a_j = \begin{bmatrix} \left(m_1^j - i_x^j m_3^j\right)^T \\ \left(m_2^j - i_y^j m_3^j\right)^T \end{bmatrix}$ et $b_j = \begin{bmatrix} i_x^j m_{34}^j - m_{14}^j \\ i_y^j m_{34}^j - m_{24}^j \end{bmatrix}$

j est la caméra (de 2 à n)

Encore une fois, l'utilisation de la pseudo-inverse de a permet de résoudre le système d'équation.

$$P = \left(a^t a\right)^{-1} a^t b \tag{2.19}$$

Cette technique possède l'avantage d'être utilisable pour un système de vision stéréoscopique binoculaire, trinoculaire ou possédant un plus grand nombre de caméras.

CHAPITRE 3

CONCEPTION ET DÉVELOPPEMENT MATÉRIEL DU CAPTEUR

Le choix du matériel est une tâche délicate puisqu'il faut tenir compte de plusieurs facteurs : les besoins technologiques du système de vision stéréoscopique, les contraintes de dimensionnement, de poids et de consommation électrique sur le robot, les ressources financières allouées au projet et finalement la disponibilité des composants.

On présente ici les trois principaux composants retenus pour le développement du prototype du capteur stéréoscopique : le laser, les caméras et l'unité de calcul.

3.1 Le laser

La section 1.3.6 présente la double contrainte contradictoire à propos du choix de la source lumineuse. Le système requiert un laser puissant pour contrer la lumière ambiante et un laser de faible intensité pour assurer la sécurité des gens. En tenant compte de ces contraintes, on peut optimiser le choix du laser et déterminer son type et sa puissance par les quatre constatations suivantes.

1. Puisque les caméras et le laser sont logés dans la tête du robot, il faut prévoir un encombrement et un poids minimum pour ceux-ci. Pour ces raisons, le choix s'est arrêté sur un laser de type diode, lequel peut offrir des dimensions et un poids considérablement réduits comparativement aux autres technologies disponibles.

2. Même si l'œil est moins sensible à la lumière infrarouge, il est important que les gens se trouvant près du robot soient conscients de la présence des faisceaux laser. C'est pourquoi l'utilisation d'un laser visible est préférable.

3. Selon la section 21 CFR 1040.10[7] du CDRH[8] régissant la réglementation pour la sécurité inhérente à l'utilisation de laser, on remarque qu'un laser de classe II n'est pas dommageable pour l'œil humain (sous certaines conditions comme le temps d'exposition devant être inférieur à ¼ de seconde et la longueur d'onde devant être dans le spectre du visible – de 400 à 700 nm). De plus, les lasers de classe II doivent avoir une puissance inférieure à 1 mW. En sachant que l'efficacité des filtres de diffraction utilisés est d'au mieux 50 % pour le premier ordre, on peut affirmer qu'un laser d'une puissance de deux fois le nombre de points émis par la matrice est sécuritaire (à raison de moins de 1 mW par faisceau). Selon le CDRH, la définition d'un laser classe II diffère de celle présentée ici car il faut prendre la mesure de la puissance totale maximale du laser à une distance de 20 cm avec une ouverture de 7 mm (correspondant à l'ouverture d'un œil humain). Si une personne se situe directement à la sortie du laser, la somme des faisceaux totalise une puissance suffisante pour endommager l'œil. Sachant qu'il est peu probable qu'une telle situation se présente, la mesure de sécurité proposée semble valable à condition évidemment d'apposer sur le robot des étiquettes indiquant le danger de la source lumineuse.

4. Puisque l'illumination de la scène est nécessaire uniquement pendant la prise d'image, il est préférable d'allumer le laser seulement à ce moment précis. Donc, pour synchroniser la lumière laser avec la prise d'image, le laser doit

---

[7] Voir : www.fda.gov/cdrh/devadvice/335.html#light
[8] Voir : www.fda.gov/cdrh

pouvoir offrir une commande de puissance. Deux avantages peuvent être tirés de cette synchronisation. Premièrement la sécurité des gens est accrue et l'énergie est économisée (un avantage important considérant les contraintes énergétiques sur un robot mobile). Deuxièmement, les besoins en puissance varient selon la puissance de la lumière ambiante. Si le laser offre un ajustement en puissance, il est possible d'ajuster la puissance de ce dernier en fonction des conditions d'éclairage instantané. La section 4.3.2.2 présente une technique permettant d'implanter cette fonction.

Le laser utilisé pour le prototype est un SNF-507X-685S-100-6.00° [9,10] de la compagnie Lasiris™ [11]. Ce laser présente tous les avantages énumérés : c'est un laser diode de 92,2 mm de longueur par 9,525 mm de rayon, de moins de 150 g, ayant une longueur d'onde de 685 nm avec une puissance variant de 0 à 100 mW pour une matrice de 7 × 7 points (qui totalise 49 points).

Figure 23 : Laser SNF™ de Lasiris™

[9] Voir : www.stockeryale.com/illumination/lasers/products/snf.pdf
[10] L'angle de diffraction du réseau n'est pas un produit standard de la compagnie Lasiris.
[11] Voir : www.lasiris.com

3.2 Les caméras

Même s'il n'y pas de contraintes reliées à la sécurité pour le choix des caméras, il existe plusieurs points à considérer. Premièrement, les caméras doivent présenter un encombrement et un poids minimums. Deuxièmement, puisque le robot peut faire des mouvements brusques et rapides, les caméras doivent faire un balayage progressif du CCD (*progressive scan*) au lieu d'un balayage par entrelacement (*interlace*) favorisant des images floues. Troisièmement, la caméra doit offrir des modes d'ajustements automatique et manuel pour le temps d'exposition et pour l'intensité (*brightness*). L'ajustement de ces paramètres permet d'implanter la technique présentée à la section 4.3.2.2. Quatrièmement, la caméra doit offrir un format d'image normalisé et accessible. Finalement, l'acquisition des images de chacune des caméras doit être synchronisée.

La caméra sélectionnée est la FireFly™ [12] de la compagnie Point Grey Research™ [13]. Cette caméra offre des avantages surprenants. Elle est basée sur un CCD Sony de 1/3" [14] à balayage progressif (*progressive scan*). Elle possède un lien numérique IEEE-1394 pour le transfert d'image et le contrôle de tous les paramètres de la caméra (incluant l'ajustement des modes automatique ou manuel du temps d'exposition et de l'intensité). Ses dimensions sont 50.8 × 50.8 × 27 mm et elle pèse moins de 100 g. Le lien numérique fait en sorte qu'aucune carte d'acquisition numérique n'est requise. Le signal reste numérique de l'acquisition jusqu'au traitement. Par ailleurs, l'alimentation de la caméra passe par le même câble. Il existe néanmoins un aspect moins intéressant pour cette petite caméra. Elle n'offre pas la possibilité de se synchroniser avec une

---

[12] Voir : www.ptgrey.com/products/firefly/index.htm
[13] Voir : www.ptgrey.com
[14] Les CCD à 1/3" offrent une plus grande surface que les CCD à 1/4". Cet avantage important leur permet de capter plus de lumière pour le même temps d'exposition.

deuxième caméra. Malgré tout, la prise de deux images peut différer de 1 ms au maximum, ce qui est suffisant pour le prototype actuel.

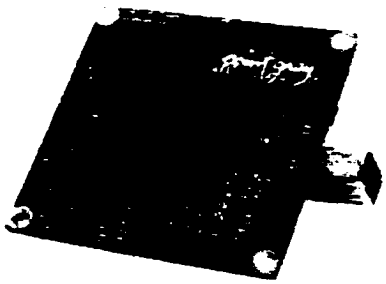


Figure 24 : Caméra Firefly™ de Point Grey Research™

Les filtres interférentiels ajoutés aux caméras ont une bande passante centrée sur la longueur d'onde du laser de 685 ± 2,5 nm.

La compagnie Point Grey Research vient de mettre en vente un tout nouveau produit, le système Bumblebee™ [15] qui semble tout à fait indiqué pour l'application en cours. C'est un système de vision stéréoscopique binoculaire complet. Ce produit est très intéressant puisqu'il offre la même souplesse au niveau du contrôle des caméras que les Firefly™ (toujours par le lien IEEE-1394). De plus, ce système est pré-calibré et il est compatible à la bibliothèque Triclops™ [16] développée par Point Grey Research (une bibliothèque de programmation dédiée à la vision stéréoscopique).

[15] Voir : www.ptgrey.com/products/bumblebee/index.htm
[16] Voir : www.ptgrey.com/products/triclopsSDK/index.htm

Figure 25 : Système de vision stéréoscopique binoculaire Bumblebee™ de Point Grey Research™

3.3 L'unité de calcul

Le choix d'une unité de calcul n'est pas simple. En fait, une solution optimale nécessite un processeur dédié avec traitement parallèle utilisant certainement des DSPs dédiés au traitement d'image. Avec un tel système informatique dédié, le système de vision serait complètement autonome et permettrait une rapidité de traitement incroyable. Malheureusement, le développement d'une telle plateforme n'est pas abordé dans ce mémoire. En fait, le développement est axé davantage sur celui d'un prototype global incluant le système de vision stéréoscopique, la représentation virtuelle de l'environnement et la planification de la trajectoire. Ainsi l'unité de calcul doit permettre une grande souplesse, une grande puissance de calcul, un encombrement et un poids réduits, au moins deux entrées de type IEEE-1394 pour interfacer les caméras précédemment sélectionnées et quelques entrées/sorties numériques et analogiques pour interfacer les périphériques externes (comme le laser).

On choisit l'ordinateur industriel 4Sight-II™ [17] de la compagnie Matrox™ [18]. Cet ordinateur industriel est très petit (184,15 × 208,28 × 83,87 mm). Il est muni d'un processeur de type Intel Celeron™ 600 MHz, 64 Mo de RAM, un disque dur de 10 Go,

[17] Voir : www.matrox.com/imaging/products/4sight2/home.cfm
[18] Voir : www.matrox.com

de trois port IEEE-1394 à 400 Mo/s, de deux ports USB, d'un port de communication Ethernet 10/100 Mo, de deux ports séries, d'un port parallèle, de sorties audio et vidéo, d'une fente d'extension PC/104-Plus™, de 16 entrées/sorties discrètes LVTTL ou opto-couplées.

Figure 26 : Unité de calcul 4Sight-II™ de Matrox™

3.4 Emplacement physique du capteur

La disposition des différents capteurs extéroceptifs a été choisie avec soin lors de la conception mécanique de Capra. Les mouvements brusques de va et vient dus à la marche du robot rendent moins efficaces les prises de mesures sur l'environnement. Par exemple, la prise d'image avec une caméra standard donne des images très floues. La solution développée ici consiste à mettre les capteurs principaux dans la tête et à fixer cette dernière sur un cou articulé pouvant la stabiliser par rapport aux mouvements du robot. Ce cou, à trois degrés de liberté, a l'avantage de stabiliser la majeure partie des capteurs extéroceptifs par rapport aux mouvements complexes du robot mais aussi de pouvoir positionner et orienter précisément ces capteurs vers une cible précise. Le lecteur peut se référer à Lagarde (1999) pour connaître les éléments de conception du cou et à Lambert (1999) pour les éléments de conception mécanique du robot.

Les deux figures suivantes montrent le schéma conceptuel du cou et sa position sur la partie supérieure avant du robot.

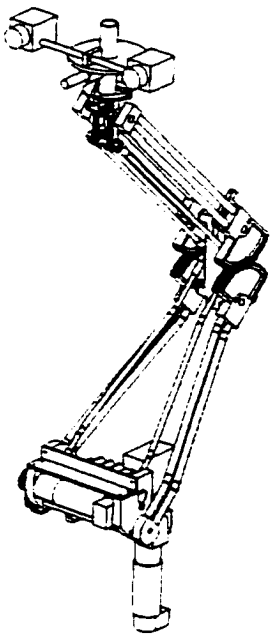


Figure 27 : Cou stabilisateur pour les capteurs extéroceptifs

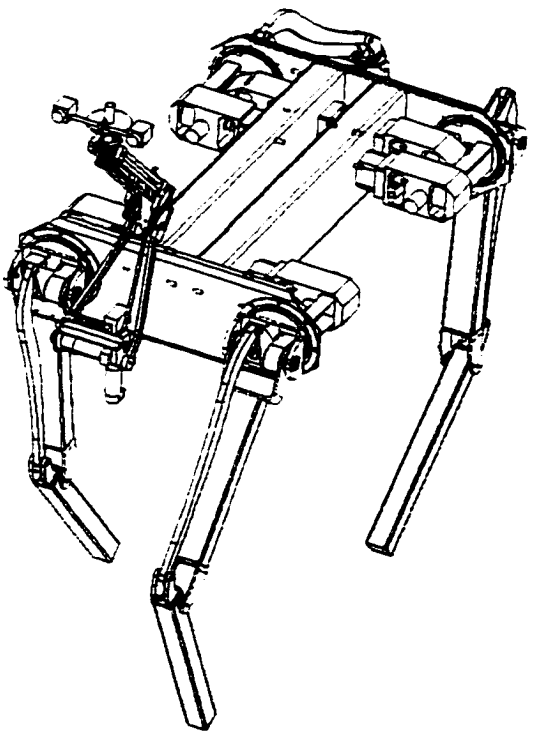


Figure 28 : Positionnement du cou sur le robot Capra

Pour définir les paramètres importants du système optique du capteur stéréoscopique, il importe de connaître plusieurs spécifications techniques du robot.

La position du capteur stéréoscopique se trouve à une hauteur nominale de 1,6 m par rapport au sol. Les contraintes de dimensions pour la tête permettent une distance maximale de 30 cm entre les caméras. Pour respecter les contraintes physiques de la structure de la tête et des caméras, ces dernières sont placées à 20 cm l'une de l'autre (20 cm entre les axes optiques). Évidemment, le laser est placé au centre des caméras de façon à ce que l'axe optique du laser soit parallèle au plan formé par les axes optiques des caméras.

Les autres aspects qu'il faut considérer pour la conception du système optique sont la surface couverte par le plan image et la matrice de points laser projetés au sol. On définit, de façon assez sommaire, que percevoir un obstacle à une distance de 3,5 m suffit pour arrêter le robot à temps. Cette estimation tient compte de la vitesse maximum du robot, du temps requis pour tout le processus algorithmique et du temps nécessaire pour que l'unité de commande puisse arrêter le robot. Finalement, on définit la largeur de la surface à observer à 2,5 m.

Ces contraintes et ces objectifs permettent de définir tous les paramètres du système optique en fonction de la position du capteur et de la surface à observer. La figure suivante montre la géométrie générale de la surface générée par le capteur stéréoscopique.

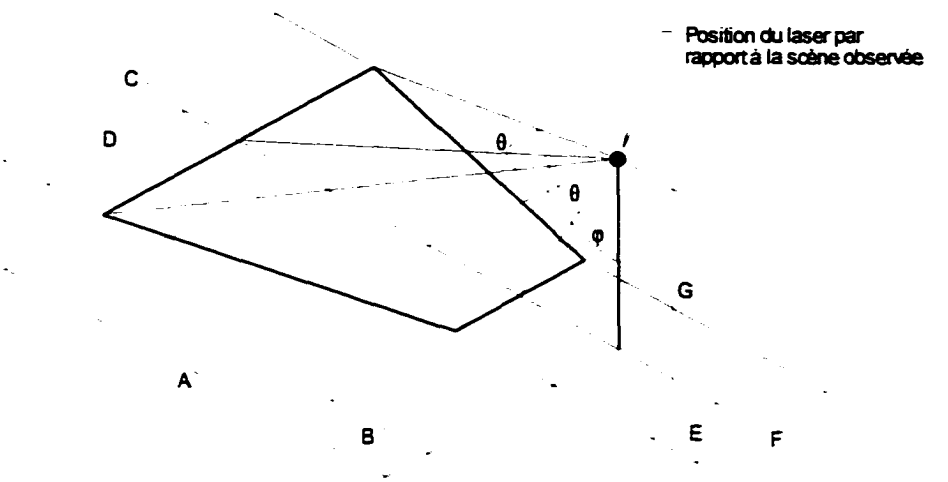


Figure 29 : Définition de l'angle total du laser et de la surface observée

Les 7 points suivants montrent le processus de définition de l'angle total défini par le laser (θ), de l'angle définissant l'orientation du capteur vers le sol (φ) et de la surface observée résultante.

1. $A = 3.5\,\text{m}$
2. $G = 1.6\,\text{m}$
3. $C = 2.5\,\text{m} \quad - \quad D = \frac{C}{2} = 1.25\,\text{m}$
4. $\theta = 2 \cdot \arctan\left(\frac{D}{\sqrt{A^2 + G^2}}\right) = 35.99°$
5. $\varphi = \arctan\left(\frac{A}{G}\right) - \frac{\theta}{2} = 47.44°$
6. $B = g \cdot \tan\left(\varphi - \frac{\theta}{2}\right) = 0.90\,\text{m}$
7. $E = \sqrt{B^2 + G^2} \cdot \tan\left(\frac{\theta}{2}\right) = 0.60\,\text{m} \quad - \quad F = 2 \cdot E = 1.19\,\text{m}$

Puisqu'on a une matrice de 7×7 points et que $\theta = 35{,}99°$, l'angle qui existe entre chaque faisceau (*interbeam angle*) est $\alpha = 6{,}00°$. Les figures 30 et 31 ainsi que le tableau suivant montrent les positions exactes de tous les points laser sur une scène où le terrain est parfaitement plat.

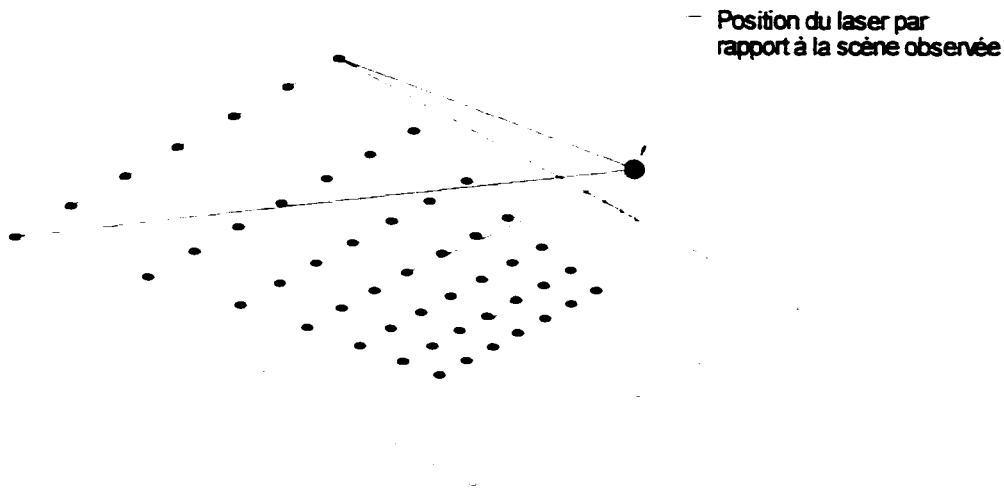


Figure 30 : Positions exactes de tous les points laser sur la scène (vue isométrique)

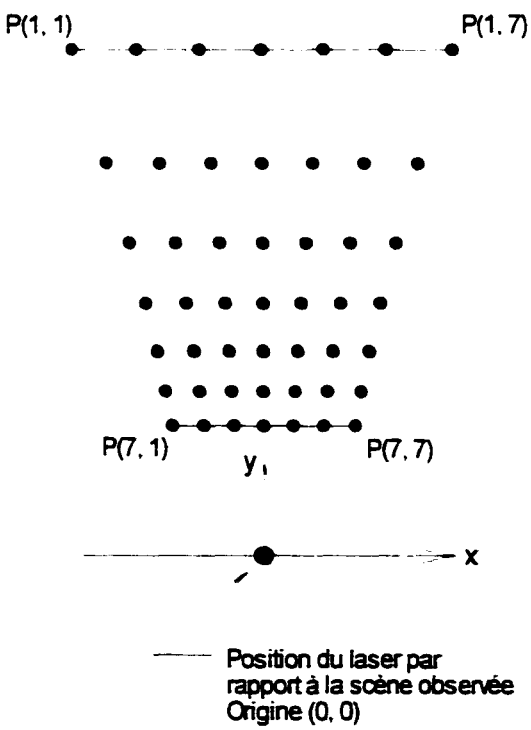


Figure 31 : Positions exactes de tous les points laser sur la scène (vue de haut)

Tableau I

Positions exactes des points laser sur la scène observée

en mètre	P(j, 1)	P(j, 2)	P(j, 3)	P(j, 4)	P(j, 5)	P(j, 6)	P(j, 7)
P(1, i)	(-1.25, 3.50)	(-0.82, 3.50)	(-0.40, 3.50)	(0, 3.50)	(0.40, 3.50)	(0.82, 3.50)	(1.25, 3.50)
P(2, i)	(-1.02, 2.71)	(-0.67, 2.71)	(-0.33, 2.71)	(0, 2.71)	(0.33, 2.71)	(0.67, 2.71)	(1.02, 2.71)
P(3, i)	(-0.87, 2.16)	(-0.57, 2.16)	(-0.28, 2.16)	(0, 2.16)	(0.28, 2.16)	(0.57, 2.16)	(0.87, 2.16)
P(4, i)	(-0.77, 1.74)	(-0.50, 1.74)	(-0.25, 1.74)	(0, 1.74)	(0.25, 1.74)	(0.50, 1.74)	(0.77, 1.74)
P(5, i)	(-0.69, 1.41)	(-0.45, 1.41)	(-0.22, 1.41)	(0, 1.41)	(0.22, 1.41)	(0.45, 1.41)	(0.69, 1.41)
P(6, i)	(-0.64, 1.14)	(-0.42, 1.14)	(-0.21, 1.14)	(0, 1.14)	(0.21, 1.14)	(0.42, 1.14)	(0.64, 1.14)
P(7, i)	(-0.60, 0.90)	(-0.39, 0.90)	(-0.19, 0.90)	(0, 0.90)	(0.19, 0.90)	(0.39, 0.90)	(0.60, 0.90)

Il est intéressant de voir maintenant quelle est la proportion de la scène que le capteur permet de voir. Si on trace les droites correspondantes aux trajets laissés par les points laser lors de l'avance du robot, on peut calculer la proportion de la scène représentée. Cette proportion varie en fonction de la résolution spatiale désirée.

Les deux figures suivantes montrent les traces laissées par l'avance et par la rotation du robot. La rotation s'effectue autour du centroïde du projecteur laser. La grille permettant d'évaluer la proportion de la surface couverte a une résolution de 10 × 10 cm.

Avec une résolution spatiale de 10 × 10 cm, la proportion couverte par le capteur est de 84 % pour l'avance du robot et de 59 % pour la rotation du robot. Il est très facile d'augmenter cette proportion sans même changer le nombre de points projetés au sol en disposant différemment le laser. En fait, il suffit de tourner le laser d'un angle d'environ 3° sur l'axe des y pour que la matrice de points s'oriente de façon à couvrir 100 % de la surface lors de l'avance du robot. D'ailleurs, on voit une lacune principale avec la configuration actuelle : le corridor central que le robot emprunte lors de l'avance reste inconnu dans une large proportion (environ 33%).

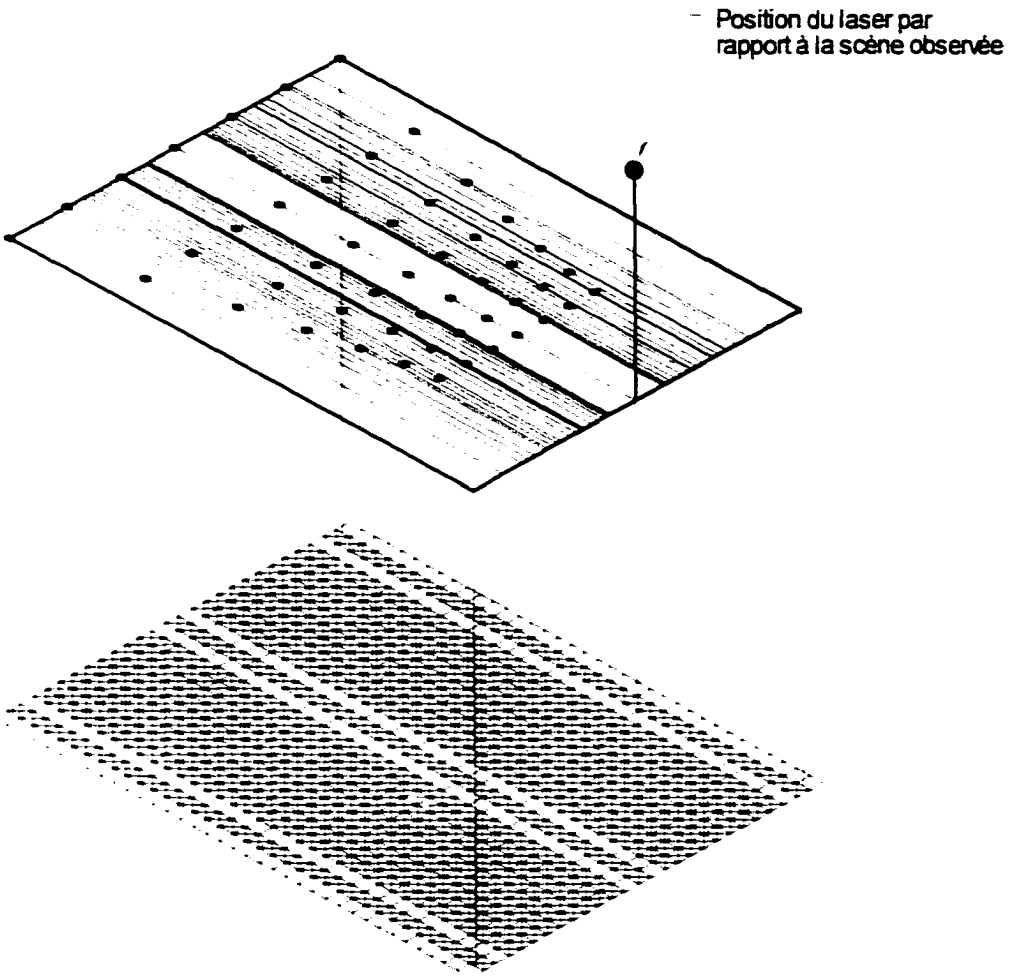


Figure 32 : Proportion de la scène modélisée lors de l'avance du robot

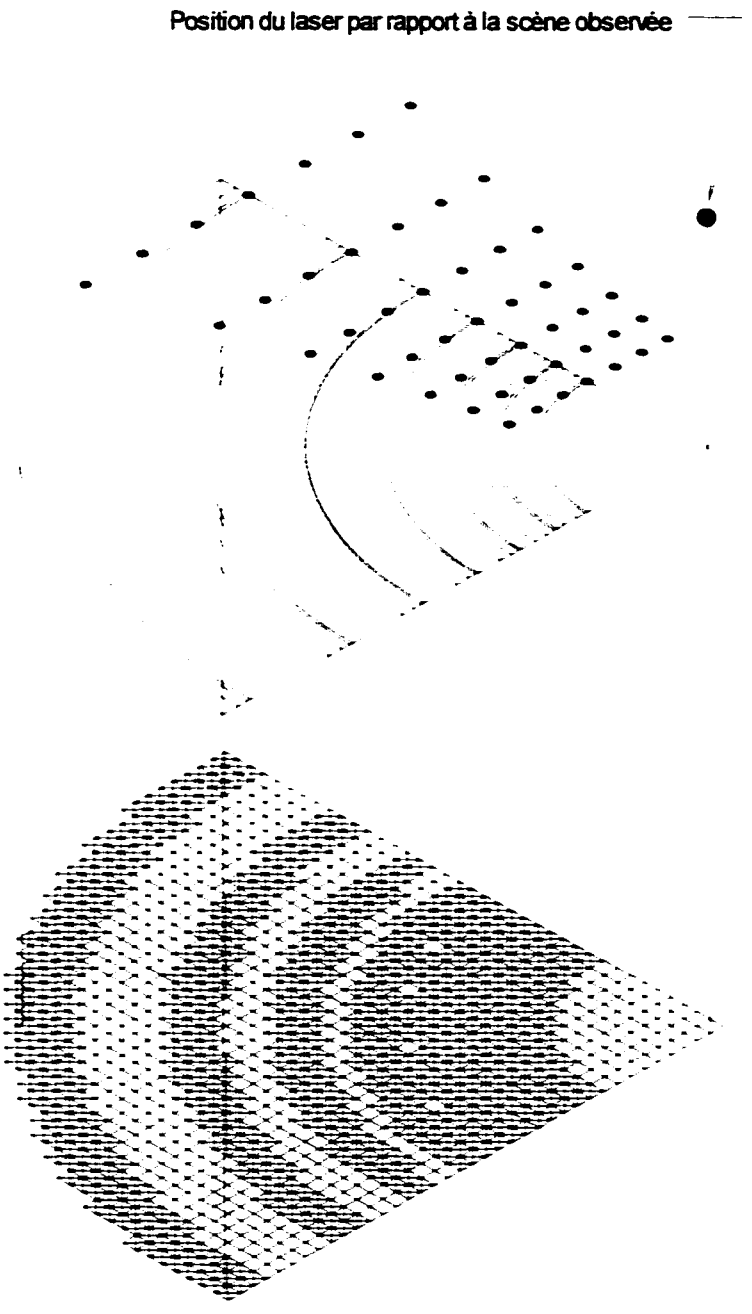


Figure 33 : Proportion de la scène modélisée lors de la rotation du robot

CHAPITRE 4

CONCEPTION ET DEVELOPPEMENT LOGICIEL DU CAPTEUR

Le processus permettant de passer de l'acquisition des images à une représentation sous forme de nuage de points est un processus traitant une somme colossale d'informations, d'où la nécessité de développer des algorithmes performants pour chaque étape du processus.

En fait, chaque étape de résolution du problème nécessite un algorithme de traitement des données permettant de simplifier le problème et de converger vers la solution. Au début du processus, les images possèdent un très grand nombre de données et constituent une représentation de très bas niveau de l'environnement. À la fin du processus, on cherche à obtenir un nombre restreint de données qui doivent constituer une représentation de plus haut niveau de l'environnement, c'est-à-dire le nuage de points.

Le processus algorithmique du capteur stéréoscopique passe par six étapes distinctes :

1. calibrage du système (étape essentielle permettant d'évaluer les matrices de projection perspective des deux caméras);
2. acquisition des images;
3. extraction des observations;
4. classification des observations;
5. appariement des observations;
6. reconstruction 3D.

La section 4.3 présente chacune des étapes de ce processus dans l'ordre mentionné.

L'étape de classification (étape 4) pose un problème intéressant. Quel type de classificateur doit-on utiliser pour distinguer les points laser du bruit? Même si ce problème semble facile à solutionner en raison de la nature circulaire des petits points laser sur l'image, il est particulièrement difficile à résoudre en considérant les scènes complexes. La section 4.1 présente la méthode utilisée pour sélectionner les descripteurs de forme les plus performants. La section 4.2 présente l'étude de trois classificateurs ainsi que la comparaison des performances de chacun. La prochaine figure montre tous les aspects algorithmiques importants.

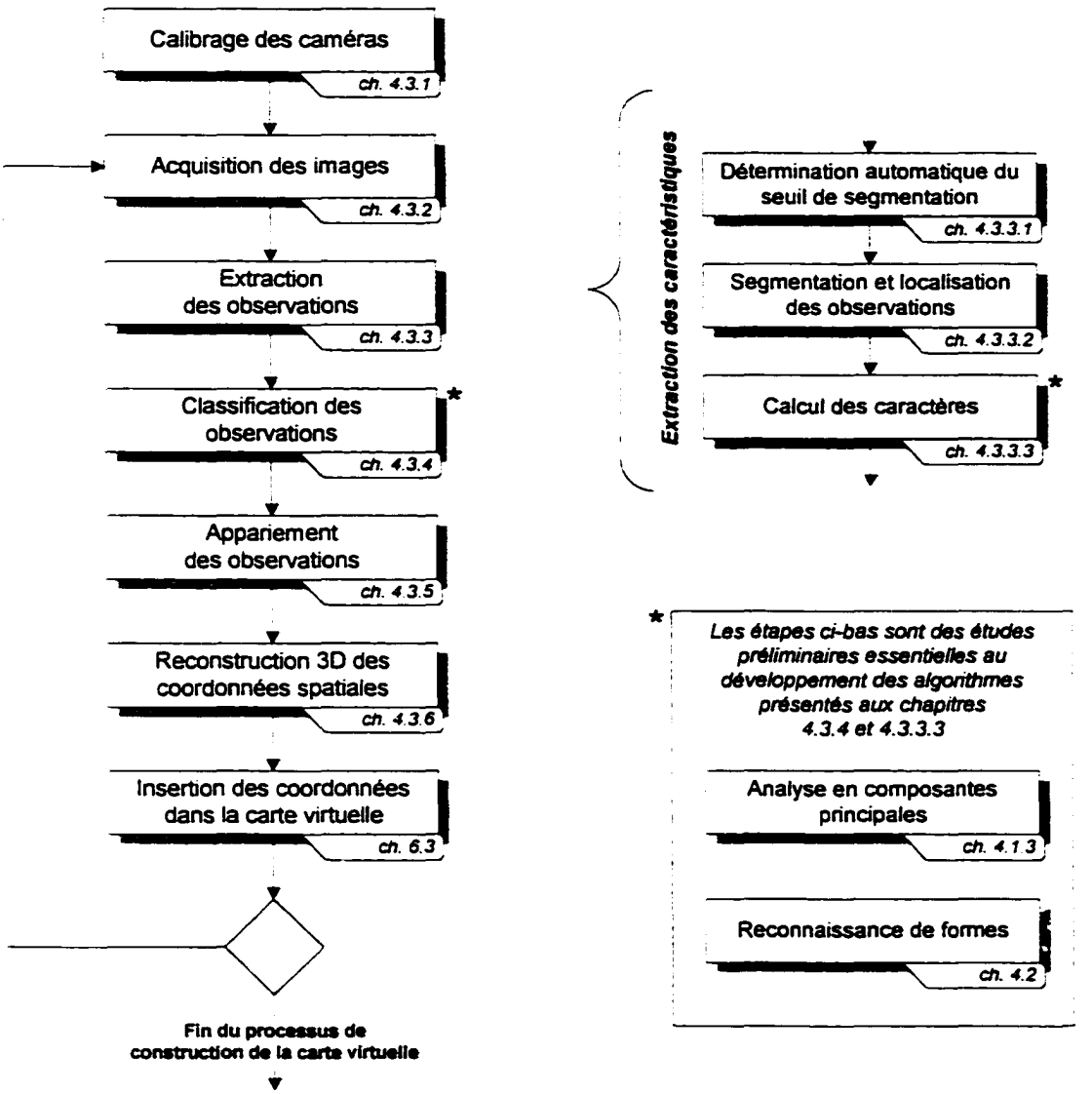


Figure 34 : Processus algorithmique du système de vision stéréoscopique

4.1 Problème de classification – Point laser ou bruit ?

Il ne suffit pas d'identifier les observations principales des images pour entamer l'opération d'appariement. Il est essentiel de déterminer quelles observations sur les images correspondent soit à un point laser, soit à du bruit. Puisque les images sont acquises dans un environnement non contrôlé, il est possible que certaines observations

parasites soient présentes sur les images. Il est ensuite inévitable que ces observation de bruit soient considérées comme des observations principales. C'est pourquoi il faut être en mesure de classifier les observations correspondant aux points laser.

Pour réussir à discriminer les deux classes observées, les points laser et les taches correspondant au bruit, il faut utiliser un classificateur. La section 4.2 couvre ce sujet mais d'abord, il importe de déterminer quels paramètres peuvent être calculés pour décrire adéquatement la forme de chacune des observations. Ces paramètres se nomment les attributs des observations.

4.1.1 Recherche et identification des attributs efficaces

Quels sont les attributs qui permettent de classifier rapidement et sans ambiguïté les observations présentes sur les images? Il existe dans la littérature plusieurs attributs pouvant être utilisés comme descripteur de formes. Idéalement, il faut utiliser des attributs simples qui puissent être calculés rapidement et qui sont invariants aux variations géométriques comme la translation, la rotation et la mise à l'échelle.

La tâche la plus difficile n'est pas de calculer la valeur numérique de ces attributs mais plutôt de choisir quels attributs doivent être utilisés pour bien décrire la forme de chaque observation. Certains peuvent être redondants et d'autres inutiles. C'est pourquoi il importe de faire une sélection précise de ces attributs. Cette section présente plusieurs attributs pouvant être utilisés ainsi que la méthode permettant de sélectionner les plus pertinents (une analyse en composantes principales).

4.1.1.1 Le contour

Le contour est une suite de pixels qui se trouvent sur le pourtour d'une observation de l'image. Il existe plusieurs façons différentes de décrire le contour mais celle présentée ici est l'une des plus utilisées. C'est la description du contour par le code de Freeman.

Le code de Freeman est une technique consistant à décrire le contour par l'identification du point d'origine et par une suite de 8 directions possibles pour chaque pixel voisin. Par convention, le point d'origine est toujours le pixel le plus en haut à gauche de l'observation. Les 8 directions indiquent les 8 voisins immédiats du pixel analysé. Chaque direction possible possède un numéro qui lui est propre. La direction vers la gauche est indiquée par le chiffre 0 et chacune des autres directions est définie en ajoutant 1 à la direction précédente en parcourant les 7 autres directions dans le sens anti-horaire.

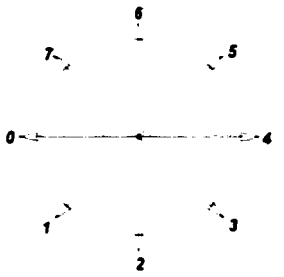


Figure 35 : Définition du code de Freeman

La figure suivante est un exemple simple permettant de bien comprendre la définition du code de Freeman.

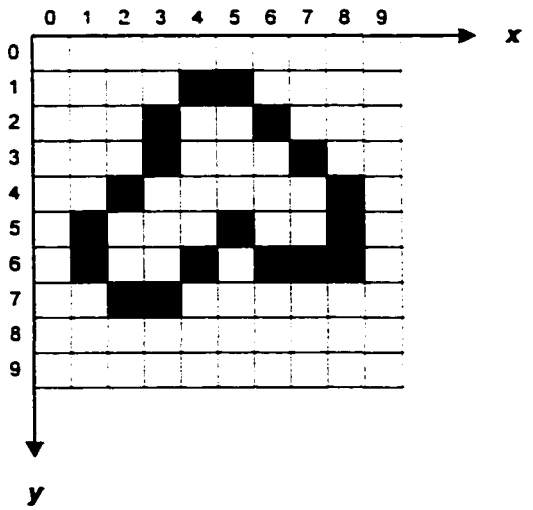


Figure 36 : Exemple du code de Freeman

Dans l'exemple illustré à la figure précédente, le point d'origine est celui identifié par une case grise se trouvant en (4, 1). Ainsi le code de Freeman associé à cet exemple est :

(4, 1) : 1-2-1-1-2-3-4-5-5-3-4-4-6-6-7-7-7-0

Il est important de comprendre dès maintenant que le contour ne peut servir comme attribut lors de la classification. Car utiliser le contour comme attribut est avantageux lorsque l'observation étudiée possède des formes géométriques particulières et spécifiques (comme le contour d'une lettre par exemple). Les points laser sont petits et possèdent une forme circulaire standard que bien d'autres attributs permettent de décrire. Néanmoins, l'évaluation du contour reste très intéressante. Premièrement, elle permet de localiser spécifiquement l'emplacement et les limites d'une observation. Deuxièmement, elle donne une représentation morphologique efficace. Finalement, elle permet, par de simples algorithmes, d'évaluer efficacement d'autres attributs.

4.1.1.2 Le nombre de pixels

Le nombre de pixels d'une observation est le nombre de pixels situés sur le contour ainsi que ceux se trouvant à l'intérieur de ce dernier.

Ce paramètre est important, car il permet de faire un premier tri sur les données. Sachant que les seules observations désirées sont des points laser, il est certain que le nombre de pixels se trouve à l'intérieur d'un intervalle fixe.

Ce premier tri permet un gain important au niveau du temps de calcul global de l'algorithme puisqu'il permet d'éviter le calcul inutile sur les observations trop petites ou trop grandes.

$$N_{pixel} = \sum_{i} \sum_{j} 1 \tag{4.1}$$

où : i et j indiquent tous les pixels composant une observation selon les axes x et y

L'intervalle du nombre de pixels est fixé empiriquement à $N_{pixel} \in [9, 125]$.

Le calcul du nombre de pixels peut se faire directement à l'aide d'une description par le code de Freeman. L'algorithme permettant de solutionner ce problème est très efficace car il nécessite simplement de parcourir les points du contour de l'observation et non tous les points intérieurs. Voici l'algorithme :

Algorithme 1 : Calcul du nombre de pixels à partir du code de Freeman

```
Soit : - N la variable contenant le nombre de pixels
       - x et y les coordonnées d'origine
       - F_i la composante du code de Freeman à l'indice i
       - x_i et y_i les coordonnées du pixel associé au code de
         Freeman F_i
       - A[0-7][0-7] = [[ 1  1  1  0  0  0  0  1]
                       [ 1  1  1  0  0  0  0  1]
                       [ 0  0  0 -1 -1 -1 -1  0]
                       [ 0  0  0  0 -1 -1 -1  0]
                       [ 0  0  0 -1 -1 -1  0  0]
                       [ 0  0  0 -1 -1 -1 -1  0]
                       [ 1  1  1  0  0  0  0  1]
                       [ 1  1  1  0  0  0  0  1]]
       - B[0-7][0-7] = [[ 0  0  0  1  1  0  0  0]
                       [ 0  0  0  1  1  1  0  0]
                       [ 0  0  0  1  1  1  1  0]
                       [ 0  0  0  0  1  1  1  1]
                       [ 1  0  0  1  1  1  0  1]
                       [ 1  1  0  1  1  1  1  1]
                       [ 0  0  0  0  0  0  0  0]
                       [ 0  0  0  1  0  0  0  0]]

1. N = y
2. Pour tous les i
3.     (x_i, y_i) = F_i
4.     N = N + A[F_(i-1)][F_i]xy_i + B[F_(i-1)][F_i]
5.     i = i + 1
```

Cet algorithme est basé sur le concept suivant : en parcourant le contour de droite à gauche, on additionne les ordonnées et en parcourant le contour de gauche à droite, on soustrait les ordonnées.

4.1.1.3 Les moments d'inertie

Les moments d'inertie d'une observation d'une image représentent la distribution de l'intensité de cette observation selon un certain référentiel.

$$M_{p,q} = \sum_{i} \sum_{j} i^p \cdot j^q \cdot I_{i,j} \tag{4.2}$$

où : $M_{p,q}$ est le moment d'inertie d'ordre p et q
i est l'abscisse
j est l'ordonnée
$I_{i,j}$ est la valeur du pixel à la coordonnée (i, j) de l'image I

4.1.1.4 L'aire

L'aire est la surface de l'observation en tenant compte de l'intensité propre à chaque pixel.

$$A = \sum_{i} \sum_{j} I_{i,j} \tag{4.3}$$

où : $I_{i,j}$ est la valeur du pixel à la coordonnée (i, j) de l'image I

On remarque que l'aire est égale au moment d'inertie $M_{0,0}$.

4.1.1.5 Le périmètre

Le périmètre est un attribut indiquant la longueur du contour. On considère la distance entre deux pixels égale à 1 si ces derniers se trouvent à une position alignée, horizontale ou verticale, l'un de l'autre et à $\sqrt{2}$ s'ils se trouvent à une position diagonale l'un de l'autre.

Le périmètre se calcule bien à l'aide du code de Freeman. Voici l'algorithme qui permet de le calculer :

Algorithme 2 : Calcul du périmètre à partir du code de Freeman

```
Soit :  - P la variable contenant la valeur du périmètre
        - Fi la composante du code de Freeman à l'indice i

1. P = 0
2. Pour tous les i
3.     si Fi = 0, 2, 4 ou 6 alors
           P = P + 1
       sinon
           P = P + √2
4.     i = i + 1
```

4.1.1.6 La complexité

La complexité est un descripteur de forme très simple et relativement performant puisqu'il est invariant à la translation, la rotation et la mise à l'échelle. Par contre, il est très sensible pour des petites formes à cause de la discrétisation du signal.

$$C = \frac{A}{P^2} \tag{4.4}$$

où : A est l'aire de la forme

P est le périmètre de la forme

4.1.1.7 Le centroïde

Le centroïde est le centre de masse de l'observation en tenant compte de la distribution de l'intensité.

$$\mu = \left(\frac{M_{1,0}}{M_{0,0}}, \frac{M_{0,1}}{M_{0,0}} \right) \tag{4.5}$$

où : μ est le centroïde

$M_{a,b}$ sont les moments d'inertie d'ordre a et b

4.1.1.8 Les moments d'inertie centrés

Ces moments sont en fait les moments d'inertie standard centrés autour du centroïde. Ils ont l'avantage d'être invariants à la translation. Ils servent principalement au calcul des moments de Pen et Keane.

$$\xi_{p,q} = \sum_i \sum_j (i - \mu_x)^p \cdot (j - \mu_y)^q \cdot I_{i,j} \tag{4.6}$$

où : $\xi_{p,q}$ est le moment d'inertie centré d'ordre p et q
i est l'abscisse et j est l'ordonnée
$I_{i,j}$ est la valeur du pixel à la coordonnée (i, j) de l'image I
μ_x et μ_y sont les valeurs x et y du centroïde

4.1.1.9 Les moments invariants de Pen et Keane

Ces moments sont invariants à la translation, la rotation et la mise à l'échelle. Ils sont des descripteurs de forme très performants.

$$\eta_{p,q} = \frac{\xi_{0,0}^{\frac{p+q}{2}+1}}{\xi_{2,0}^{\frac{p+1}{2}} \cdot \xi_{0,2}^{\frac{q+1}{2}}} \cdot \xi_{p,q} \tag{4.7}$$

où : $\eta_{p,q}$ est le moment de Pen et Keane d'ordre p et q
$\xi_{p,q}$ est le moment d'inertie centré d'ordre p et q

4.1.1.10 La saturation

La saturation est un paramètre qu'on introduit pour tenir compte de l'intensité relative des points laser par rapport à une autre référence. Pour avoir les résultats les plus significatifs, l'intensité du point laser doit être comparée au fond de l'image se trouvant à proximité du point laser. Avec une telle comparaison, le critère de saturation est indépendant des conditions d'éclairage et de la surface sur laquelle est projeté le faisceau laser. Malheureusement, il est très difficile d'évaluer le fond de l'image avec précision

autour d'une observation puisque les discontinuités locales génèrent des variations d'éclairage. Ainsi la complexité que peut prendre la forme d'une observation nécessite une méthode élaborée pour déterminer l'intensité du fond. Cette méthode n'est pas développée.

La technique implantée est beaucoup plus simple et calcule le ratio en fonction de l'intensité maximale que le point peut avoir si la caméra est saturée.

$$S = \frac{A}{N_{pixel} \cdot I_{max}} \tag{4.8}$$

où : S est l'indice de saturation

A est l'aire

N_{pixel} est le nombre de pixels

I_{max} est l'intensité maximum d'un pixel (typiquement $I_{max} = 2^n - 1 = 255$)

4.1.1.11 La ressemblance avec une distribution normale bivariante

On introduit ici un autre attribut qui apporte une nouvelle estimation intéressante pour l'identification d'une observation correspondant à un point laser. Cette caractéristique est certainement la plus intéressante d'un point de vue théorique puisqu'elle est basée sur une caractéristique inhérente à la lumière laser. En effet, la distribution énergétique du faisceau laser selon une coupe transversale correspond exactement à une distribution normale bivariante. Même si un laser standard (He-Ne) possède un profil symétrique, le type de laser utilisé pour le robot (laser diode) possède un profil non symétrique de type ellipsoïde. L'équation mathématique d'une telle distribution est

$$d(x,y)=\frac{1}{\sqrt{2\pi}\sigma_x\sigma_y}\cdot e^{-\frac{1}{2}\left(\left(\frac{x-\mu_x}{\sigma_x}\right)^2+\left(\frac{y-\mu_y}{\sigma_y}\right)^2\right)} \qquad (4.9)$$

où : $d(x, y)$ est la fonction de distribution normale bivariante à la position (x, y)

μ_x et σ_x sont la moyenne et l'écart-type de la distribution suivant l'axe des X

μ_y et σ_y sont la moyenne et l'écart-type de la distribution suivant l'axe des Y

La figure suivante illustre la distribution énergétique typique d'une coupe d'un faisceau laser. Le phénomène de distribution ellipsoïdale est lié à la nature même de la diode laser. C'est la dimension très réduite de la jonction P-N à l'intérieur de la diode qui produit un effet de diffraction et qui dilate un axe par rapport à l'autre. Mathématiquement, la distribution énergétique sur l'axe le plus long est représentée par l'écart-type le plus grand.

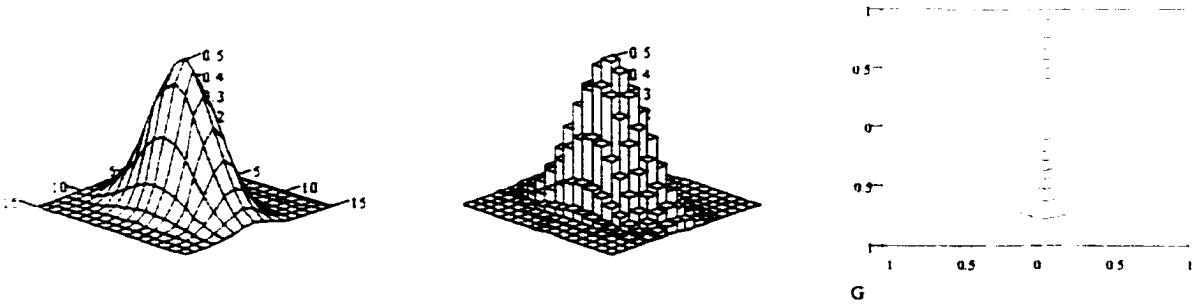


Figure 37 : Distribution énergétique d'un faisceau laser selon une coupe transversale

Pour pouvoir se servir de cette caractéristique, il suffit de calculer la ressemblance existant entre l'observation et la distribution normale bivariante correspondante. Pour ce faire, on calcule la surface de distribution normale bivariante qui épouse le mieux le point image. La méthode des moindres carrés est utilisée pour optimiser ce « *curve fitting* ».

De plus, puisque l'amplitude de la surface est fonction de σ, on ajoute à l'équation 4.9 un facteur d'amplitude α qui permet d'obtenir une amplitude correspondant à celle des données sans normaliser ces dernières, ce qui donne

$$D(x,y)=\frac{\alpha}{\sqrt{2\pi}\sigma_x\sigma_y}\cdot e^{-\frac{1}{2}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2+\left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]} \tag{4.10}$$

où : $D(x, y)$ est la fonction de distribution normale bivariante modifiée
α est le facteur d'amplitude

On établit dès maintenant la fonction d'erreur quadratique permettant d'évaluer la différence entre la courbe D et le point analysé.

$$E(x,y)=\sum_i\sum_j\left(\frac{\alpha}{\sqrt{2\pi}\sigma_x\sigma_y}\cdot e^{-\frac{1}{2}\left[\left(\frac{i-\mu_x}{\sigma_x}\right)^2+\left(\frac{j-\mu_y}{\sigma_y}\right)^2\right]}-I_{i,j}\right)^2 \tag{4.11}$$

où : $E(x, y)$ est la fonction d'erreur quadratique
$I_{i,j}$ est l'intensité de l'image à la position (i, j)

Il faut maintenant minimiser l'erreur quadratique pour l'ensemble des points de l'image. Il est tentant de résoudre cette équation de façon analytique par la technique des dérivées partielles et en mettant ces dernières égales à zéro (ce qui donne cinq équations, $\frac{\partial E}{\partial \alpha}=0$, $\frac{\partial E}{\partial \mu_x}=0$, $\frac{\partial E}{\partial \mu_y}=0$, $\frac{\partial E}{\partial \sigma_x}=0$ et $\frac{\partial E}{\partial \sigma_y}=0$ avec cinq inconnus α, μ_x, μ_y, σ_x et σ_y). Malheureusement, il est impossible de faire les manipulations adéquates pour solutionner analytiquement un tel problème. Il faut utiliser une méthode de résolution numérique.

L'emploi d'une méthode de résolution numérique est mal approprié dans le contexte d'une application temps réel. Les besoins en temps réel du robot nécessitent une

méthode convergeant vers la solution en un temps toujours égal, ce qui est difficile à réaliser avec une méthode numérique. Par contre, il est possible de simplifier suffisamment le problème pour revenir à une solution analytique. En fait, les moyennes μ_x et μ_y sont déjà connues car elles correspondent exactement au centroïde du point. Ainsi le nombre de paramètres est réduit à trois. Cette réduction est encore insuffisante car il n'existe toujours pas de solution analytique.

Il est possible de poser l'hypothèse suivante permettant de réduire le nombre de paramètres à un seul. Même si la nature asymétrique de la distribution énergétique est bien réelle, l'optique mise à la sortie du laser servant à faire le focus du faisceau laser réduit la différence entre les deux axes de façon telle qu'on peut les supposer identiques. En fait, la prise de mesure rend cette hypothèse vérifiable puisque les distances séparant les points laser des caméras sont grandes. Ainsi on obtient $\sigma_x = \sigma_y = \sigma$.

De plus, il existe une relation étroite entre la dimension d'un point laser et l'écart-type σ. On détermine empiriquement que $\sigma \approx 2/9 \cdot d$ (d étant la dimension de l'observation). Avec ces hypothèses, l'équation à cinq inconnus est réduite à un seul inconnu, c'est-à-dire α. Il est maintenant possible de trouver une solution analytique à ce problème d'optimisation. On cherche donc la solution de $\frac{\partial E}{\partial \alpha} = 0$.

$$\frac{\partial E}{\partial \alpha} = \frac{\partial}{\partial \alpha}\left(\sum_i \sum_j \left(\frac{\alpha}{\sqrt{2\pi}\sigma_x \sigma_y} \cdot e^{-\frac{1}{2}\left(\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2 \right)} - I_{i,j} \right)^2 \right) = 0$$

$$\sum_i \sum_j \left(\frac{\alpha}{\sqrt{2\pi}\sigma_x \sigma_y} \cdot e^{-\frac{1}{2}\left(\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2 \right)} - I_{i,j} \right) \cdot \frac{2}{\sqrt{2\pi}\sigma_x \sigma_y} \cdot e^{-\frac{1}{2}\left(\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2 \right)} = 0$$

$$\alpha \cdot \sum_i \sum_j \left(\frac{1}{\sqrt{2\pi}\sigma_x \sigma_y} \cdot e^{-\frac{1}{2}\left[\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2\right]} \right)^2 = \sum_i \sum_j \frac{2I_{i,j}}{\sqrt{2\pi}\sigma_x \sigma_y} \cdot e^{-\frac{1}{2}\left[\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2\right]}$$

$$\alpha = 2\sqrt{2\pi}\sigma_x \sigma_y \frac{\sum_i \sum_j I_{i,j} \cdot e^{-\frac{1}{2}\left[\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2\right]}}{\sum_i \sum_j \left(e^{-\frac{1}{2}\left[\left(\frac{i-\mu_x}{\sigma_x}\right)^2 + \left(\frac{j-\mu_y}{\sigma_y}\right)^2\right]} \right)^2} \tag{4.12}$$

Pouvant maintenant évaluer la valeur de α, μ_x, μ_y, σ_x et σ_y de façon à minimiser l'erreur quadratique, il est possible de calculer l'indice de ressemblance existant entre la courbe de distribution normale bivariante et la matrice de données correspondant au point image. L'indice de corrélation r est donné par :

$$r = \sqrt{1 - \frac{\sum_i \sum_j \left(I_{i,j} - D_{i,j}\right)^2}{\sum_i \sum_j \left(I_{i,j} - \bar{I}\right)^2}} \tag{4.13}$$

où : r est l'indice de corrélation

$I_{i,j}$ est le pixel de l'image à la position (i, j)

$D_{i,j}$ est la valeur de la distribution normale bivariante à la position (i, j)

$\bar{I} = \frac{\sum_i \sum_j I_{i,j}}{w^2}$ correspond au niveau d'intensité moyenne de la portion d'image analysée et w est la dimension de l'observation analysée

Le coefficient r ne peut être calculé simplement avec les données brutes disponibles. Le profil du point laser est différent du profil de distribution normale bivariante puisque ce dernier contient aussi l'intensité du fond de la scène. Ainsi il faut réussir à soustraire au

signal ce *bruit* uniforme qu'est le fond. Comme pour la saturation, le problème reste difficile à solutionner puisque la valeur du fond est difficile à évaluer à cause des discontinuités locales potentielles.

La figure suivante est un exemple des données brutes des images acquises par le robot (à gauche) et de la soustraction du fond sur cette même image (à droite).

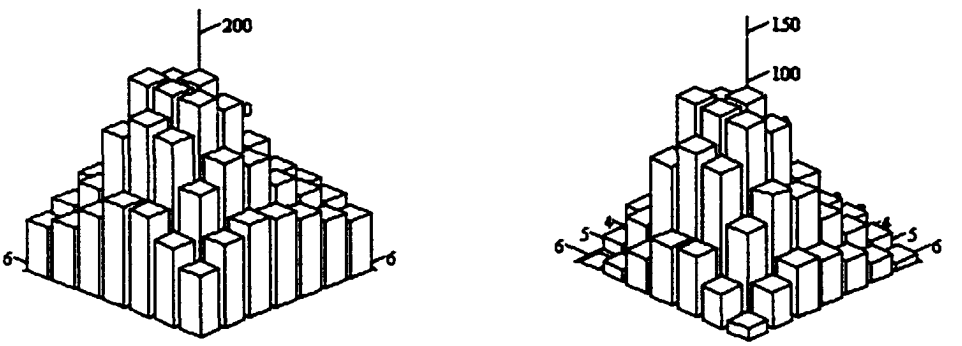


Figure 38 : Images d'un point laser avec et sans le fond de la scène

La méthode implantée et illustrée ci-dessus permet un traitement acceptable et surtout rapide. Elle consiste simplement à évaluer la valeur du fond par une moyenne de tous les pixels se trouvant sur le pourtour de l'image. Il suffit ensuite de soustraire cette valeur à l'image initiale.

Malheureusement l'approximation de l'intensité de fond, la discrétisation du signal et l'hypothèse où σ est établi selon une règle empirique rendent le calcul de corrélation parfois impossible. En effet, avec ces approximations, le ratio calculé est parfois supérieur à 1. Par exemple le fond évalué peut être d'une intensité supérieure à quelques pixels se trouvant à l'intérieur de la zone d'intérêt. Pour contrer cette limitation, on modifie le calcul de corrélation par un calcul de ressemblance de la façon suivante :

$$r' = \sqrt{\frac{\sum_i \sum_j (I_{i,j} - D_{i,j})^2}{\sum_i \sum_j (I_{i,j} - \bar{I})^2}} \qquad (4.14)$$

où : r' est l'indice de ressemblance

L'interprétation de ce nouvel indice est fort simple : plus il est près de 0, plus le point image représente une distribution normale bivariante. Pour les raisons évoquées précédemment, la valeur peut être supérieure à 1. Il ne faut pas se surprendre par contre que certains points laser possèdent un très mauvais taux de ressemblance. En effet, plus le point est petit, plus le calcul de l'indice de ressemblance est approximatif et ce à cause de la discrétisation du signal.

Le tableau suivant montre quelques exemples qui illustrent le calcul de l'attribut de ressemblance avec une distribution normale bivariante.

Tableau II

Exemples de ressemblance avec une distribution normale bivariante

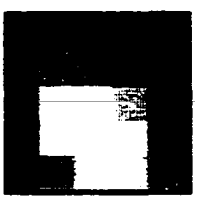
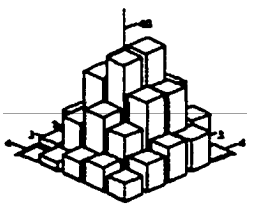
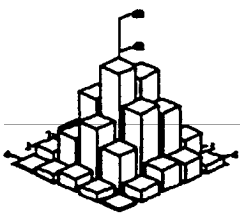
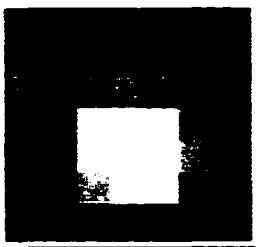
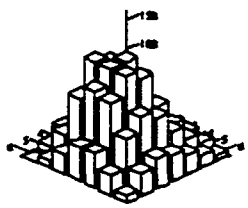
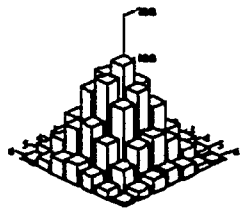
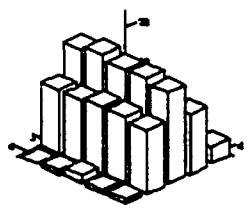
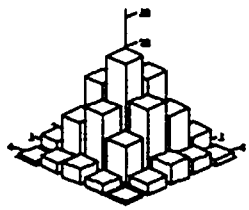
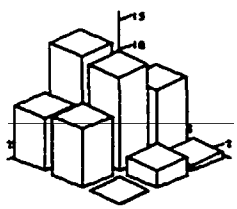
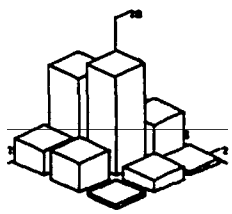
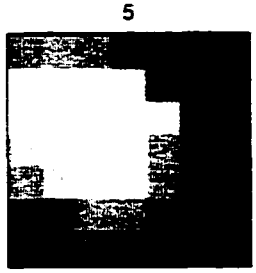
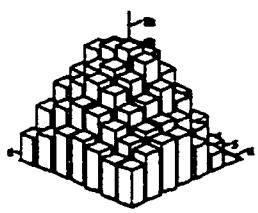
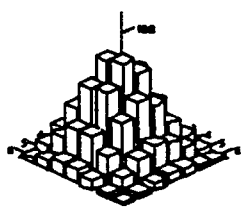
Image	Distribution de l'image	Distribution normale bivariante	Paramètres
1 			Dimension = 5 α = 196,46 μ_x = 1,71 μ_y = 2,05 $\sigma_x = \sigma_y$ = 10/9 r = 0,928 r' = 0,371

Tableau II (suite)

Image	Distribution de l'image	Distribution normale bivariante	Paramètres
2 			Dimension = 7 α = 126,3 μ_x = 3,22 μ_y = 2,95 $\sigma_x = \sigma_y$ = 14/9 r = 0,863 r' = 0,506
3 			Dimension = 5 α = 104,02 μ_x = 1,85 μ_y = 1,94 $\sigma_x = \sigma_y$ = 10/9 r = 0,42 r' = 0,908
4 			Dimension = 3 α = 27,91 μ_x = 0,961 μ_y = 0,584 $\sigma_x = \sigma_y$ = 2/3 r = 0,852 r' = 0,524
5 			Dimension = 7 α = 27,91 μ_x = 2,921 μ_y = 2,589 $\sigma_x = \sigma_y$ = 14/9 r = *erreur* r' = 1,299

Le cinquième exemple du tableau précédent montre un cas où l'estimation du fond de l'image est erronée. En fait, l'évaluation de l'intensité moyenne du fond de l'image dans cette région est surestimée. Ainsi après soustraction du fond, les coins sombres de

l'image se retrouvent inférieur à 0. C'est pourquoi l'indice de corrélation r et l'indice de ressemblance r' ont ces valeurs.

Pour le reste du document, Ψ fait référence à l'indice de ressemblance avec un modèle de distribution normale bivariante.

4.1.2 Construction de la base de données

Maintenant que les descripteurs de formes sont identifiés, on construit une base de données possédant un grand nombre d'observations permettant l'analyse de ces attributs. Cette section présente les méthodes utilisées et les outils développés pour réaliser la base de données. Celle-ci sert à deux étapes : l'analyse en composantes principales permettant de déterminer quels descripteurs de forme utiliser et le développement du classificateur. La construction de la base de données est une tâche fastidieuse nécessitant une minutie particulière. En effet, il faut identifier plusieurs milliers d'observations présentes sur de nombreuses images. De plus, il faut déterminer la classe d'appartenance et calculer les attributs choisis comme descripteur de forme pour chaque observation.

Une première base de données comportant près de 1 800 observations est réalisée. Les prototypes de certains classificateurs permettent de constater que cette base de données contient trop peu d'informations et qu'au moins 4 000 observations sont nécessaires (idéalement jusqu'à 12 000 observations!). Une image parfaite et sans bruit possède une cinquantaine d'observations. Par contre, on peut compter jusqu'à 120 observations sur les images les plus bruitées (on se réfère à des observations dont le nombre de pixels est inclus dans l'intervalle défini – voir 4.1.1.2). Selon les images prises, on estime à 65 en moyenne le nombre d'observations. Ainsi pour construire une base de données comportant 4 000 observations, 65 images environ sont nécessaires. La taille de la base de données résultante est considérable puisque pour chaque observation, on doit conserver les 147 attributs nécessaires à l'analyse en composantes principales et

conserver la classe d'appartenance. La définition de la classe d'appartenance de chaque observation est essentielle pour l'élaboration des classificateurs.

Un logiciel dédié à la construction de la base de données est développé pour minimiser les chances d'erreur et pour maximiser le temps requis pour la construire. Ce logiciel, développé en langage C++ est constitué d'une interface graphique optimisée pour l'identification de la classe de chaque observation faite sur l'image. Le moteur d'analyse d'images est intégré à ce logiciel pour identifier automatiquement les observations et pour calculer la valeur numérique des descripteurs de formes. Les données peuvent être affichées à l'écran ou sauvegardées dans un fichier texte pouvant être repris par n'importe quel logiciel (notamment MathCad™ et MatLab™ utilisés pour l'analyse en composantes principales et le développement du classificateur). Le classement d'une observation se fait simplement en cliquant sur l'observation. De cette façon, la classe associée à une observation passe de *Acceptée*, *Ambiguë*, *Rejetée* et ainsi de suite.

Voici les descriptions des différentes parties du logiciel présenté à la figure 39 :

1. Onglet affichant l'image ou les informations numériques de la dernière observation sélectionnée.
2. Affichage de l'image analysée.
3. Les différentes observations sont classées en trois catégories : acceptée (en vert), ambiguë (en mauve) ou rejetée (en rouge). L'usager n'a qu'à cliquer sur une observation pour changer la catégorie de cette dernière.
4. Histogramme de luminance et seuil de segmentation.
5. Boutons de commande.
6. Données sur la dernière observation cliquée.
7. Onglet affichant le tableau des données.
8. Tableau des données. Les lignes correspondent aux observations et les colonnes aux attributs. Quelques colonnes sont ajoutées pour afficher l'information de quelques calculs intermédiaires.
9. Choix de l'ordre des moments à afficher (ordre p et q pour $M_{p,q}$, $\xi_{p,q}$ et $\eta_{p,q}$).

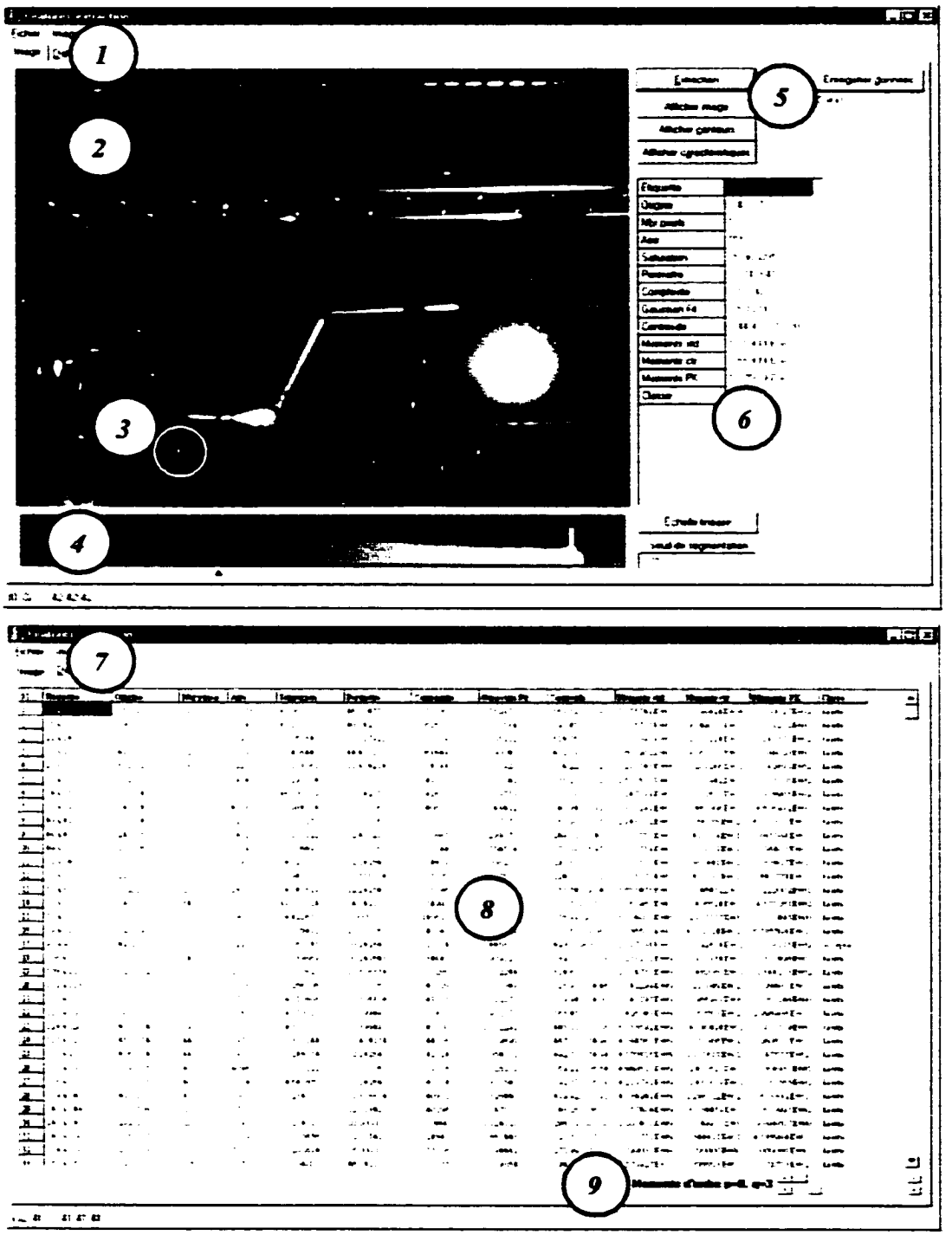


Figure 39 : Interface usager du logiciel développé pour la construction de la base de données

Lors de la construction de la base de données, il est important d'avoir un maximum d'observations décrivant tous les cas possibles. La prise d'image doit être une étape méticuleusement préparée permettant d'acquérir un nombre comparable d'observations pour tous les cas possibles. Pour la création de la base de données, plusieurs scènes représentant divers environnements communs sont utilisées. Les images obtenues de ces scènes sont peu bruitées et, de cette façon, il est impossible d'obtenir des observations de tous les cas possibles. C'est pourquoi il faut se donner la peine de créer des environnements complexes produisant des images problématiques à résoudre. Ces images permettent d'obtenir un grand nombre d'observations de toutes sortes. La base de données utilisée dans ce mémoire est créée à partir de cinq scènes dont une seule représente un environnement simple. Toutes les autres scènes représentent des environnements plus ou moins complexes. La figure suivante montre une photo prise de la scène la plus complexe utilisée pour la prise d'image. À gauche, on voit une photo normale prise de la scène et à droite l'image telle que vue par le système de vision. La scène est constituée d'un chariot fait de tiges métalliques très spéculaires et de deux sources lumineuses dont une est directement orientée vers le système de vision. De plus, une source de lumière blanche très puissante sert d'éclairage ambiant et permet de voir les éléments de la scène malgré l'usage des filtres interférentiels.

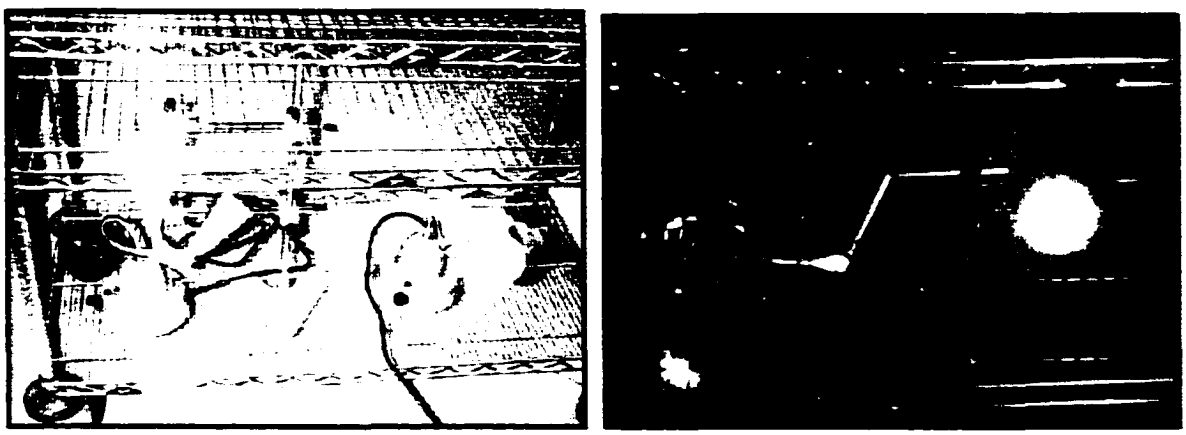


Figure 40 : Scène complexe ayant servie à la construction de la base de données

En fait cette scène est si complexe que les images de plusieurs réflexions spéculaires sont pratiquement identiques aux images des points laser.

Ainsi une centaine d'images décrivant différents points de vue des cinq scènes sont minutieusement analysées à l'aide du logiciel précédemment décrit. La tâche est difficile pour la personne effectuant le classement des observations car elle ne doit pas se laisser influencer par ses connaissances de la scène. Étant des réflexions spéculaires, certains observations sont identiques aux points laser et doivent être classées comme étant des points laser. Ce faux classement est essentiel car aucune technique basée sur la reconnaissance de forme ne peut classer adéquatement ce type d'observation. La tâche est pratiquement impossible sans une analyse complexe de la scène et une compréhension complète des constituants de la scène. C'est pourquoi la connaissance et l'analyse du contexte de la scène ne doivent pas influencer le jugement de la personne qui fait la classification.

La base de données réalisée totalise pas moins de 6 000 observations. Ces 6 000 observations sont classées en trois classes et possèdent chacune près de 2 000 observations. Plus exactement, on a 2 150 observations de points laser, 1 800 observations ambiguës et 2 050 observations de bruit.

4.1.3 Sélection des attributs : analyse en composantes principales

Si on considère tous les attributs énumérés, on remarque que certains sont fonction d'autres attributs. Ainsi certains attributs n'apportent aucune connaissance supplémentaire sur les observations. C'est le cas notamment pour le nombre de pixels ainsi que l'aire et le périmètre qui servent aux calculs de la complexité et de la saturation.

D'un autre côté, il est difficile d'évaluer si un attribut apporte une contribution significative à la description d'une observation. C'est le cas notamment des moments de

Pen et Keane. Il est possible que le moment d'ordre $\eta_{a,b}$ apporte une réelle contribution à la description de l'observation contrairement au moment d'ordre $\eta_{c,d}$.

L'analyse en composantes principales est une technique permettant de représenter des données selon certains critères optimaux. Ces critères algébriques et géométriques permettent de faire une projection du nuage de points sur les hyperplans qui maximisent l'étalement des données. Ainsi on peut passer d'un espace R^N à un espace R^M (où $M < N$) tout en minimisant la perte d'information sur les données. Cette optimisation peut être réalisée en maximisant la distance entre chaque donnée lors de la projection. Ainsi par le choix judicieux d'un hyperplan qui minimise la perte d'information, l'analyse en composantes principales permet de réduire le nombre d'attributs nécessaires à la description d'une observation. Le lecteur intéressé peut se référer à Bouroche et Saporta (1980) et Lebart, Morineau et Piron (2000).

Avant de commencer l'analyse des données, il faut construire une base de données possédant un grand nombre d'observations avec, pour chacune d'elle, le calcul de tous les attributs qui doivent être analysés. La construction d'une telle base de données est décrite à la section précédente.

Même si la base de données construite possède un grand nombre d'attributs, seuls quelques uns sont utilisés pour l'analyse en composantes principales. Certains attributs peuvent être éliminés dès maintenant puisqu'ils n'apportent pas d'information pertinente ou supplémentaire sur la description d'une observation. C'est le cas pour le nombre de pixels, l'aire, le périmètre, le centroïde et les moments d'inertie (centrés ou non). Les attributs retenus sont la complexité, la saturation, la ressemblance avec un modèle de distribution normale bivariante et les moments de Pen et Keane variant de 0 à 11. Tous ces attributs totalisent, avec les 144 moments de Pen et Keane, pas moins de 147 attributs à analyser. Il est évident que seulement quelques uns de ces moments suffisent

à décrire convenablement une observation. Le choix d'aller jusqu'au moment d'ordre 11×11 est arbitraire.

On pose maintenant R la matrice correspondant à la base de données. Chaque ligne de cette matrice est reliée à une observation et chaque colonne à un attribut.

Il est important de bien définir la métrique utilisée pour calculer la distance entre chaque observation. En considérant seulement les données brutes, il est difficile de déterminer une façon efficace pour calculer la distance puisque chacun des attributs s'exprime sur des échelles différentes et avec des distributions différentes. Si on désire utiliser une formule aussi simple que celle de Pythagore pour calculer la distance, on doit centrer et réduire les données. La métrique utilisée est la suivante : les données sont centrées en soustrayant de chaque observation le centroïde du nuage de point.

$$\overline{R}_j = R_j - \bar{r} \tag{4.15}$$

où : R est la matrice des données et $\overline{R}$ est la matrice des données centrées

$\bar{r}$ est le vecteur du centroïde du nuage de points et se calcule par $\bar{r}_j = \frac{1}{n}\sum_{i=1}^{n} R_{ij}$

n est le nombre de données

Pour réduire les données, on divise par l'écart-type les données centrées. Cette métrique possède deux avantages. Premièrement, la distance entre deux individus est sans dimension, donc elle ne dépend plus des unités impliquées. Deuxièmement, cette métrique donne une importance égale à tous les attributs quel que soit leur dispersion.

$$\tilde{R}_{ij} = \frac{\bar{R}_{ij}}{s_j} \tag{4.16}$$

où : $\bar{R}$ est la matrice des données centrées et $\tilde{R}$ est la matrice des données centrées réduites

s est le vecteur représentant l'écart-type des attributs : $s_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\bar{R}_{ij} - \bar{r}_j\right)^2}$

La métrique s'écrit alors comme étant la matrice M suivante :

$$M = \begin{bmatrix} 1/s_1 & 0 & \dots & 0 \\ 0 & 1/s_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1/s_n \end{bmatrix} \tag{4.17}$$

La distance euclidienne qui existe entre deux observations se calcule ainsi :

$$d(a.b) = \sqrt{\left(\tilde{R}_a - \tilde{R}_b\right)^T \left(\tilde{R}_a - \tilde{R}_b\right)} \tag{4.18}$$

où : $d(a,b)$ est la distance entre les observations a et b

La métrique étant correctement définie, on calcule les valeurs propres de la matrice de variance-covariance multipliée par la métrique.

$$\lambda = \phi\left(\overline{R}^T \cdot P \cdot \overline{R} \cdot M\right) \tag{4.19}$$

où : λ est le vecteur des valeurs propres des attributs

ϕ est l'opérateur qui calcule les valeurs propres d'une matrice

$\overline{R}$ est la matrice des données centrées

P est la matrice de pondération définie par $P = \begin{bmatrix} 1/m & 0 & \dots & 0 \\ 0 & 1/m & \dots & 0 \\ \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & 1/m \end{bmatrix}$

m étant le nombre d'attributs

$\overline{R}^T \cdot P \cdot \overline{R}$ donne la matrice de variance-covariance des attributs

M est la métrique précédemment définie

λ représente aussi l'inertie expliquée par les axes factoriels qui peuvent être décrits par les vecteurs propres de $\overline{R}^T \cdot P \cdot \overline{R} \cdot M$. On calcule τ correspondant à λ normalisé (en pourcentage) et T correspondant à τ cumulatif par

$$\tau_i = 100 \cdot \frac{\lambda_i}{\sum \lambda} \tag{4.20}$$

$$\mathrm{T} = \begin{cases} \mathrm{T}_1 = \tau_1 \\ \mathrm{T}_k = \tau_k + \mathrm{T}_{k-1} \quad \text{pour k de 2 à m} \end{cases} \tag{4.21}$$

où : τ correspondant à λ normalisé

T correspondant à τ cumulatif

Alors, en classant par ordre décroissant le vecteur T, on voit apparaître dans l'ordre les attributs les plus significatifs à la description des données.

Les résultats obtenus à partir des données recueillies sont les suivants :

Tableau III

Distribution de l'inertie totale expliquée par les 12 attributs les plus significatifs

	Ψ	$\eta_{0,2}$	$\eta_{0,1}$	$\eta_{0,6}$	$\eta_{1,1}$	$\eta_{0,0}$	$\eta_{0,5}$	*Sat.*	$\eta_{0,4}$	$\eta_{0,3}$	*Comp.*	$\eta_{1,0}$	...
τ	24,13	21,21	15,01	11,15	7,89	5,63	5,06	3,47	2,08	1,87	0,54	0,36	...
T	24,13	45,34	60,35	71,50	79,39	85,02	90,08	93,55	95,63	97,50	98,04	98,40	...

Le tableau précédent montre que 12 attributs suffisent à décrire 98,4 % de l'inertie totale du nuage de données. En d'autres termes, chacune des observations peut être suffisamment bien décrite par seulement 12 attributs, ce qui diminue de beaucoup l'ensemble des attributs du début.

Le choix d'utiliser 12 attributs est tout à fait arbitraire. Si une description à 95 % de l'inertie totale suffit, alors seulement 9 attributs peuvent être utilisés. Pour la même raison, 15 attributs sont nécessaires pour une description à 99 %.

Les attributs choisis sont les suivants :

$$\Psi - \textbf{Saturation} - \textbf{Complexité}$$

$$\eta_{0,0} - \eta_{0,1} - \eta_{1,0} - \eta_{1,1} - \eta_{0,2} - \eta_{0,3} - \eta_{0,4} - \eta_{0,5} - \eta_{0,6}$$

Il est possible pour le classificateur de prendre une décision automatique et de classer les différentes observations à partir des 12 descripteurs de formes trouvés. Évidemment, pour chaque observation trouvée sur l'image, il faut calculer la valeur numérique de ces attributs.

4.2 Reconnaissance de formes

Maintenant que les descripteurs de formes sont judicieusement sélectionnés, on aborde le vrai problème de classification. À première vue, il peut sembler facile de classer une

observation. La différence existant entre la description d'un point laser et d'un bruit est souvent très grande. Ainsi la conception d'un classificateur peut paraître simple. En fait, il n'en est rien. Certains bruits sont tellement semblables aux points laser que même les classificateurs les plus performants ne peuvent les classifier. C'est le cas notamment des bruits issus d'une réflexion très spéculaire, comme ceux créés par la réflexion d'une source lumineuse autre que le laser sur une surface polie.

La figure suivante présente une image où se trouvent des cas d'une très grande ambiguïté. On peut voir quatre observations agrandies. Les observations 1 et 2 correspondent à des points laser tandis que les observations 3 et 4 correspondent à du bruit. En regardant l'image, il peut sembler facile d'identifier quelles observations sont des points laser et lesquelles sont du bruit. Mais cette évidence vient seulement du fait qu'il est possible de voir le contenu de la scène et qu'une personne solutionne le problème en faisant une analyse contextuelle des éléments de l'image. Si la scène est invisible, personne ne peut discriminer les deux types d'observations.

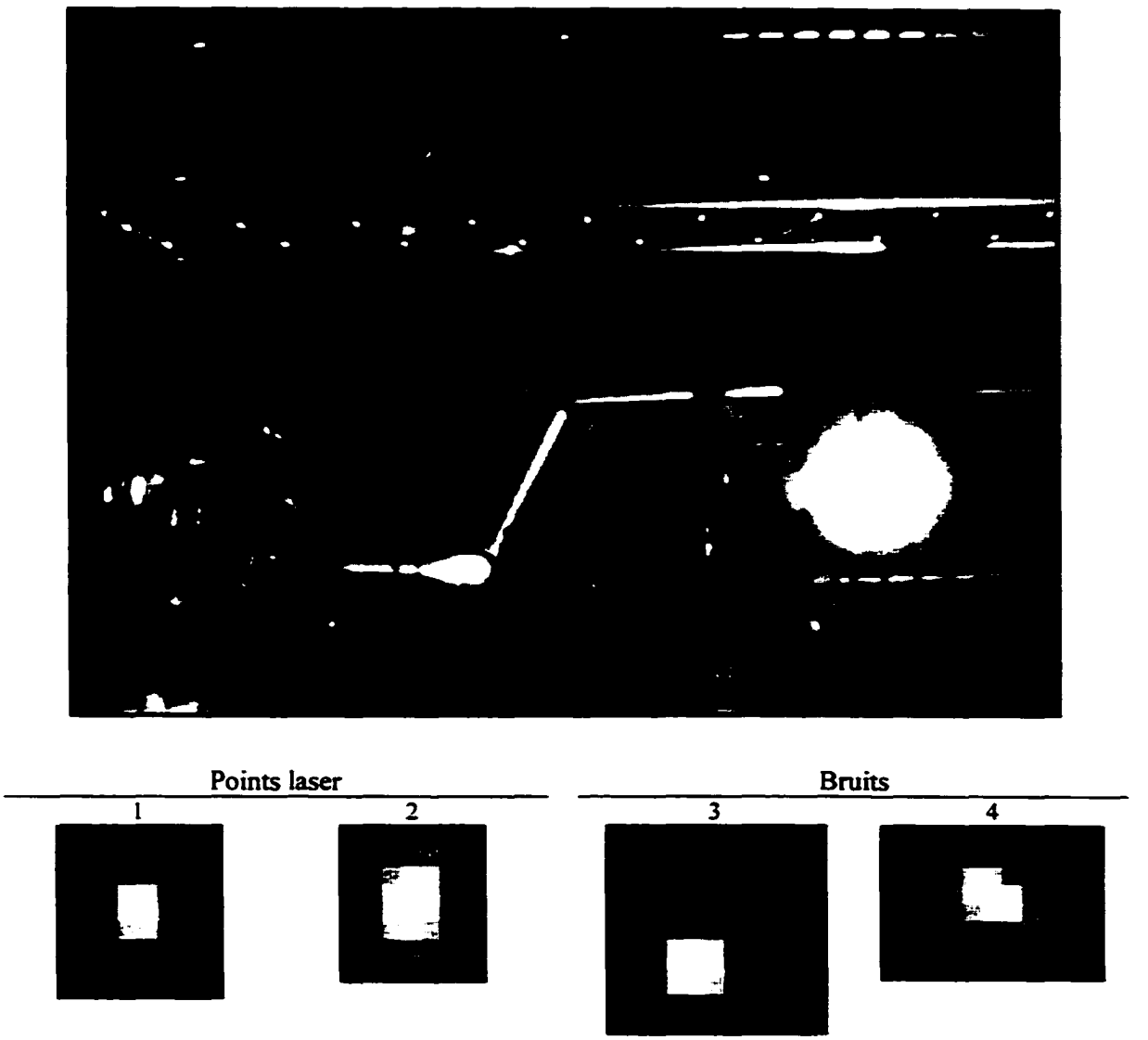


Figure 41 : Image ayant des observations ambiguës

Pour solutionner cet épineux problème, trois classificateurs sont mis à l'essai : le Bayes quadratique, le *K-NN* et un réseau de neurones de type perceptron multicouches.

4.2.1 Optimisation des paramètres et méthode de comparaison

Pour réaliser chacun des classificateurs présentés, l'ensemble de données est premièrement divisé en deux sous-ensembles. Le premier sert à l'apprentissage et le

second est utilisé pour les tests de performance du classificateur, c'est-à-dire pour évaluer l'erreur de généralisation. Si un seul ensemble est utilisé pour l'apprentissage et les tests, il est certain que l'évaluation de l'erreur de généralisation est biaisée. Par contre, en utilisant deux groupes distincts, l'erreur de généralisation ne peut être biaisée.

Certains algorithmes de reconnaissance de forme possèdent des paramètres de contrôle dont la valeur optimale est inconnue et dépend de la nature de l'ensemble de données. Pour s'en convaincre, il suffit de penser au nombre de voisin dans le *K-NN* ou au nombre de neurones se trouvant sur la couche cachée dans un réseau de neurones. Il n'est pas très pertinent de comparer différents classificateurs sans optimiser leurs paramètres puisqu'un classificateur peut avoir ses paramètres beaucoup plus près de leurs valeurs optimales qu'un autre lors de l'initialisation. Ainsi toute comparaison reste inutile sans cette optimisation. La technique utilisée pour optimiser les paramètres consiste à faire plusieurs apprentissages en faisant varier les paramètres à optimiser.

L'ensemble de données utilisé pour l'optimisation ne doit pas servir comme ensemble d'apprentissage ou comme ensemble servant à évaluer l'erreur de généralisation. On doit simplement créer un troisième sous-ensemble pour faire cette tâche. Ainsi l'ensemble de données initiales est divisé en trois parties, la première servant à l'optimisation des paramètres, la seconde servant à l'apprentissage et la dernière servant à évaluer l'erreur de généralisation non biaisée.

La technique de la validation croisée est utilisée pour augmenter la précision sur la mesure de l'erreur de généralisation. Cette technique consiste simplement à faire plusieurs séries de test en mélangeant les données pour chacun des sous-ensembles. Une validation croisée basée sur un mélange de 10 sous-groupes est faite pour chaque classificateur.

La figure suivante montre l'organisation de la base de données. La taille des groupes est définie par la taille de la classe ayant le moins d'observations. Ainsi il existe trois

classes ayant chacune trois ensembles de 600 observations. Chacun de ces ensembles est divisé en 10 sous-ensembles de 60 observations.

On peut se demander pourquoi définir une troisième classe alors que seulement deux classes suffisent à la classification. Une première base de données avec seulement deux classes a été réalisée. une classe pour les points laser et une classe pour les autres observations. Aucun classificateur n'a pu atteindre des performances acceptables. atteignant parfois jusqu'à un taux d'erreur de classification de 40 %. La raison est que certaines observations sont particulièrement difficiles à classer et que les deux classes se recoupent grandement dans l'espace R^{12} des attributs. L'utilisation d'une troisième classe permet de scinder l'espace des attributs en trois groupes qui se recoupent beaucoup plus légèrement. De cette façon. la nouvelle classe correspondant aux observations ambiguës recoupe la classe des points laser et celle du bruit. Par contre. il n'y a pratiquement aucun recoupement entre les deux autres classes. Même si le classificateur ne peut classer adéquatement certaines observations. le problème est limité à la classe des observations ambiguës. En acceptant seulement les observations des points laser. l'erreur de classification est considérablement diminuée.

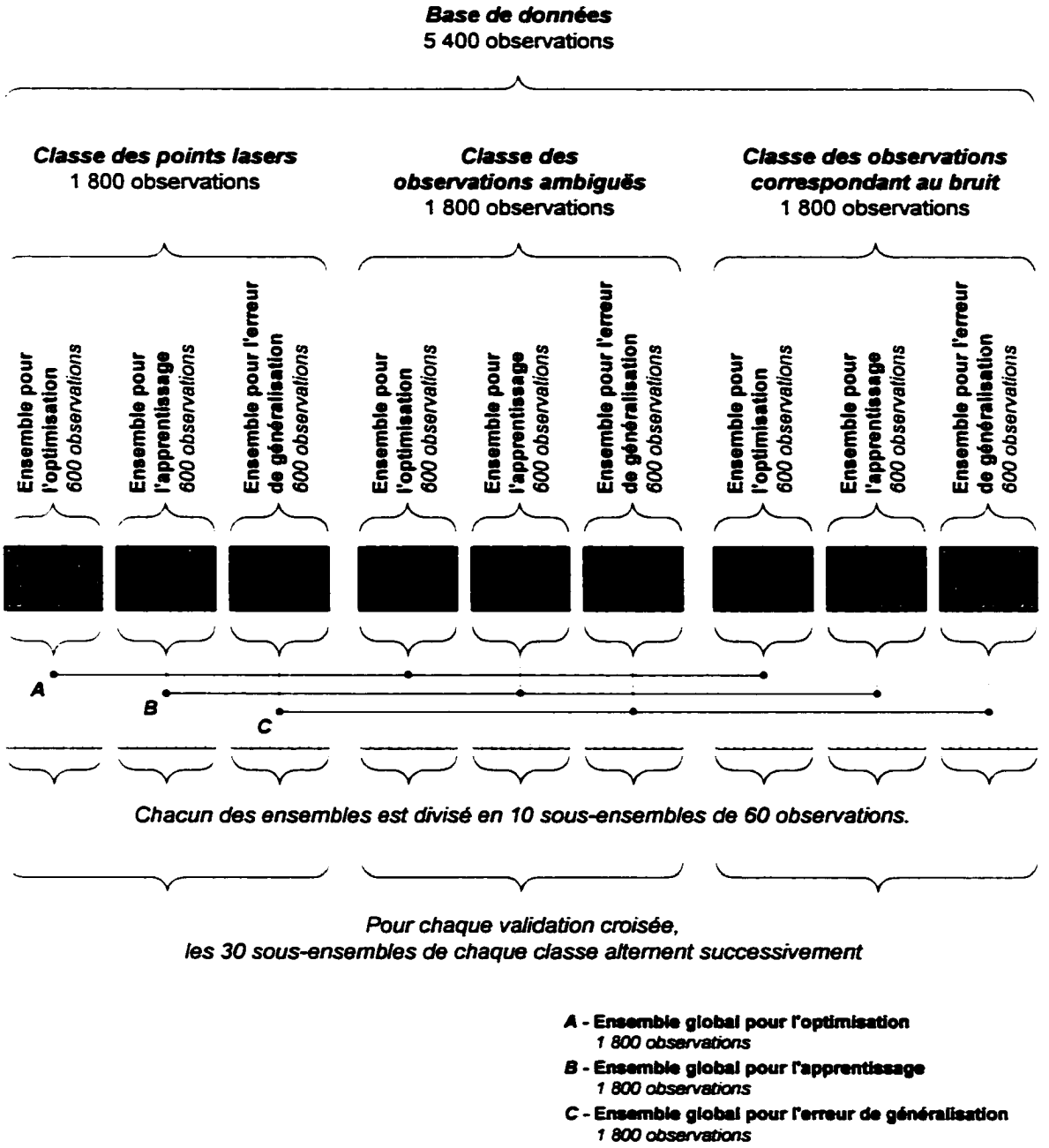


Figure 42 : Organisation de la base de données

Les classificateurs sont comparés selon deux critères : la performance de classification et le temps requis pour faire la classification. Il est essentiel pour une application aussi exigeante en terme de temps de calcul de considérer dans le choix d'un classificateur la

rapidité à laquelle il peut classer une observation. À titre de référence, l'ordinateur utilisé pour les calculs est de type Intel Pentium III™, 750 MHz, 256 Mo de RAM avec, comme système d'opération, Windows 2000™ et comme outil de calcul, MatLab 5.3™.

4.2.2 Classificateur de type Bayes quadratique

Le classificateur Bayes quadratique est un classificateur paramétrique basé sur la méthode statistique bayesienne. La règle de Bayes est définie ainsi

$$p(x) = \frac{p(x|c_i) \cdot P(c_i)}{\sum_{j=1}^{k} p(x|c_j) \cdot P(c_j)} \tag{4.22}$$

On peut calculer une distance statistique entre l'observation x et la classe c_i (i = 1, 2 ou 3) de la façon suivante :

$$d_i = \ln(P(c_i)) + \frac{1}{2}\ln\left(\left|\hat{S}_i\right|\right) + \frac{1}{2}(x - \bar{x}_i)^t \cdot \hat{S}_i^{-1} \cdot (x - \bar{x}_i) \tag{4.23}$$

où : c_i est la classe i

$P(c_i)$ est la probabilité à priori. $P(c_i) = \frac{1}{M}$. M est le nombre de classe

$\bar{x}_i$ est le vecteur de la moyenne des observations de la classe i

x est le vecteur de l'observation à classer

$\hat{S}_i$ est la matrice de covariance des observations de la classe i

$\left|\hat{S}_i\right|$ est le déterminant de la matrice $\hat{S}_i$

Cette méthode de classification ne contient aucun paramètre à optimiser. Malgré le fait que pour ce classificateur il n'est pas nécessaire de scinder la base de données en trois, l'expérimentation est faite selon l'organisation présentée à la figure 42. De cette façon,

la comparaison entre les trois classificateurs est plus cohérente. Le tableau suivant montre les résultats obtenus avec ce classificateur.

Tableau IV

Résultats obtenus avec le classificateur de Bayes

Validation croisée	**Erreur de généralisation non biaisée**		**Temps moyen de classification pour une observation** - ms -
	Erreur totale (sur les 3 classes) - % -	***Erreur sur la classe des points laser*** - % -	
Ensemble test 1	17,17	6,50	2,0000
Ensemble test 2	16,33	5,50	2,1875
Ensemble test 3	16,50	5,67	2,1875
Ensemble test 4	16,33	4,67	2,1938
Ensemble test 5	14,50	4,83	2,1313
Ensemble test 6	14,00	5,17	2,1875
Ensemble test 7	16,17	5,00	2,2563
Ensemble test 8	14,83	5,50	2,1875
Ensemble test 9	16,67	6,00	2,1875
Ensemble test 10	14,50	5,17	2,1938
Moyenne	15,70	5,40	2,1713
Écart-type	1,12	0,56	0,0671

Le résultat pour le critère de classification réussie est assez mauvais. Un taux de bonne classification correspondant à une erreur de 5,40 % a des conséquences néfastes sur tout le processus algorithmique. Si on prend une image complexe à analyser et que 120 observations sont faites sur l'image, il y a environ six observations mal classées et considérées comme des points laser. Par la suite, ces erreurs se répercutent dans tout le processus algorithmique et créent quelques faux appariements causant l'insertion de points virtuels inexistants. Dans le pire des cas, on peut considérer que jusqu'à 6 faux points peuvent être ajoutés à la carte virtuelle et que 5 % de la représentation virtuelle de l'environnement est erroné.

D'un autre côté, le temps requis pour la classification est très court.

4.2.3 Classificateur de type *K* plus proches voisins (*K-NN*)

La technique des *k* plus proches voisins est basée sur le concept de distance dans l'espace des attributs. Cette technique simple consiste à calculer la distance existant entre l'observation désirée et toutes les observations de référence (soit les ensembles d'apprentissage de la base de données). On peut alors classer une observation en prenant la classe des *k* observations les plus près.

On utilise deux méthodes pour calculer la distance entre deux observations : la distance euclidienne et la distance de Mahalanobis. Respectivement, ces distances se calculent ainsi :

$$d_i^2 = (x - o_i)'(x - o_i) \tag{4.24}$$

$$\delta_i^2 = (x - o_i)' S^{-1} (x - o_i) \tag{4.25}$$

où : d_i^2 est le carré de la distance séparant l'observation o_i et l'observation x

S est la matrice de covariance

Pour éviter des calculs longs et inutiles, on omet la racine carrée dans les calculs de distance. Ainsi on compare le carré de la distance entre chaque observation. Les résultats obtenus avec les deux distances sont très similaires. Avec la distance de Mahalanobis, on observe une augmentation du taux de bon classement de 0,63 %. Par contre, le temps requis pour ce calcul est de deux fois supérieur au calcul de la distance euclidienne. La petite amélioration de performance qu'offre la distance de Mahalanobis ne justifie pas le temps de calcul qu'elle requiert. C'est la raison pour laquelle la distance euclidienne est utilisée.

On ajoute à l'algorithme le paramètre k_{min}. Ce paramètre supplémentaire indique le nombre minimum d'observations requises de la même classe parmi k paramètres les plus près pour déterminer la classe d'appartenance. Donc $k_{min} < k$ et donne l'avantage d'éviter certains classements ambigus. Dans le même ordre d'idée, s'il existe deux classes ayant le même nombre d'observations les plus proches, on rejette le classement. Cette contrainte supplémentaire peut devenir obsolète avec certains agencements de k et k_{min}.

Il existe deux paramètres à optimiser pour cet algorithme de classement, k et k_{min}. Pour faire cette optimisation, on fait varier k de 3 à 35 et k_{min} de 2 à k par incrément de 1. La figure suivante illustre les résultats obtenus. On remarque que plus k augmente, plus le nombre d'erreurs augmente. Il est difficile de déterminer les valeurs optimales puisqu'il n'existe pas de minimum sur le graphique. On détermine alors que si k vaut 10, il n'y a aucune erreur de classement tout en ayant un nombre intéressant d'observations qui sont comparées. De plus, avec k_{min} qui vaut 7, ce paramètre reste sévère tout en permettant quelques observations différentes.

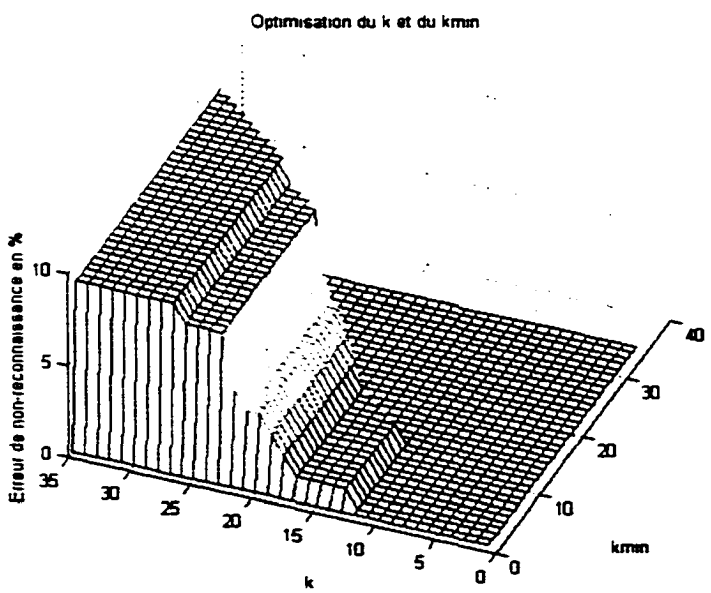


Figure 43 : Optimisation des paramètres k et k_{min}

Le tableau suivant montre les résultats obtenus selon ce classificateur avec les paramètres $k = 10$ et $k_{min} = 7$.

Tableau V

Résultats obtenus avec le classificateur *K-NN*

Validation croisée	**Erreur de généralisation non biaisée**		**Temps moyen de classification pour une observation** - ms -
	Erreur totale (sur les 3 classes) - % -	***Erreur sur la classe des points laser*** - % -	
Ensemble test 1	43,83	0,00	141,750
Ensemble test 2	38,50	0,50	140,250
Ensemble test 3	42,17	0,17	141,000
Ensemble test 4	41,67	0,33	141,750
Ensemble test 5	43,33	0,00	141,750
Ensemble test 6	41,17	0,00	140,500
Ensemble test 7	39,83	0,17	142,250
Ensemble test 8	43,17	0,50	141,750
Ensemble test 9	42,50	0,00	141,750
Ensemble test 10	39,17	0,17	144,250
Moyenne	41,53	0,18	141,700
Écart-type	1,84	0,20	1,098

La différence entre l'erreur totale et l'erreur sur la classe des points laser est considérable. Ceci permet de constater que la disposition spatiale dans R^{12} des trois classes de données est très entrecoupée. Les erreurs de classification viennent du fait que plusieurs points étant classés comme des observations ambiguës, sont en fait des observations de points laser ou de bruit. Par contre, il y a très peu de mauvais classement pour les deux autres classes. Par cet exemple flagrant, on voit pourquoi il est pertinent, voire essentiel, d'ajouter une troisième classe pour la classification. L'erreur de généralisation obtenue pour la classe des points laser montre des résultats très intéressants. Une erreur de classification de l'ordre de 0,18 % est plus que suffisante si on considère que la plupart des images montrent peu d'observations complexes à

résoudre. Par contre, le temps de calcul requis pour chaque classification est beaucoup trop long. Si un processeur semblable à celui utilisé ici est dédié à la tâche de classification, seulement sept observations peuvent être classifiées par seconde, ce qui est tout à fait inacceptable.

Il est possible de réduire considérablement la taille de l'ensemble d'apprentissage contenant plusieurs observations très similaires. Il faut donc optimiser un troisième hyper-paramètre, celui de la taille de l'ensemble d'apprentissage. Cette optimisation peut se faire facilement en faisant varier la taille de l'ensemble du maximum (600 dans le cas ici) jusqu'à ce que la performance du classificateur varie. Pour optimiser la diversité de l'observation contenue dans l'ensemble d'apprentissage, il suffit de calculer la distance entre chaque point et tous les autres. De cette matrice de distance, il suffit d'éliminer ceux qui offrent les distances les plus petites. Ce mémoire ne présente pas le résultat d'une telle optimisation.

4.2.4 Classificateur de type réseau de neurones

Les motivations pour utiliser un réseau de neurones sont reliées à sa capacité de classifier des classes non linéairement séparables et à sa grande rapidité à réaliser les classifications. Le réseau de neurones utilisé est un réseau de type perceptron multicouches avec, comme technique d'apprentissage, la méthode de rétro-propagation du gradient d'erreurs. Le réseau est limité à une seule couche cachée.

L'architecture du réseau de neurones développé est très simple. On y retrouve 12 neurones à l'entrée pour les 12 attributs sélectionnés, n neurones sur la couche cachée (où n est un paramètre à optimiser) et 3 neurones de sortie représentant chacune des classes. La figure suivante montre cette architecture ainsi que le nom des variables et des indices utilisés pour les définitions mathématiques.

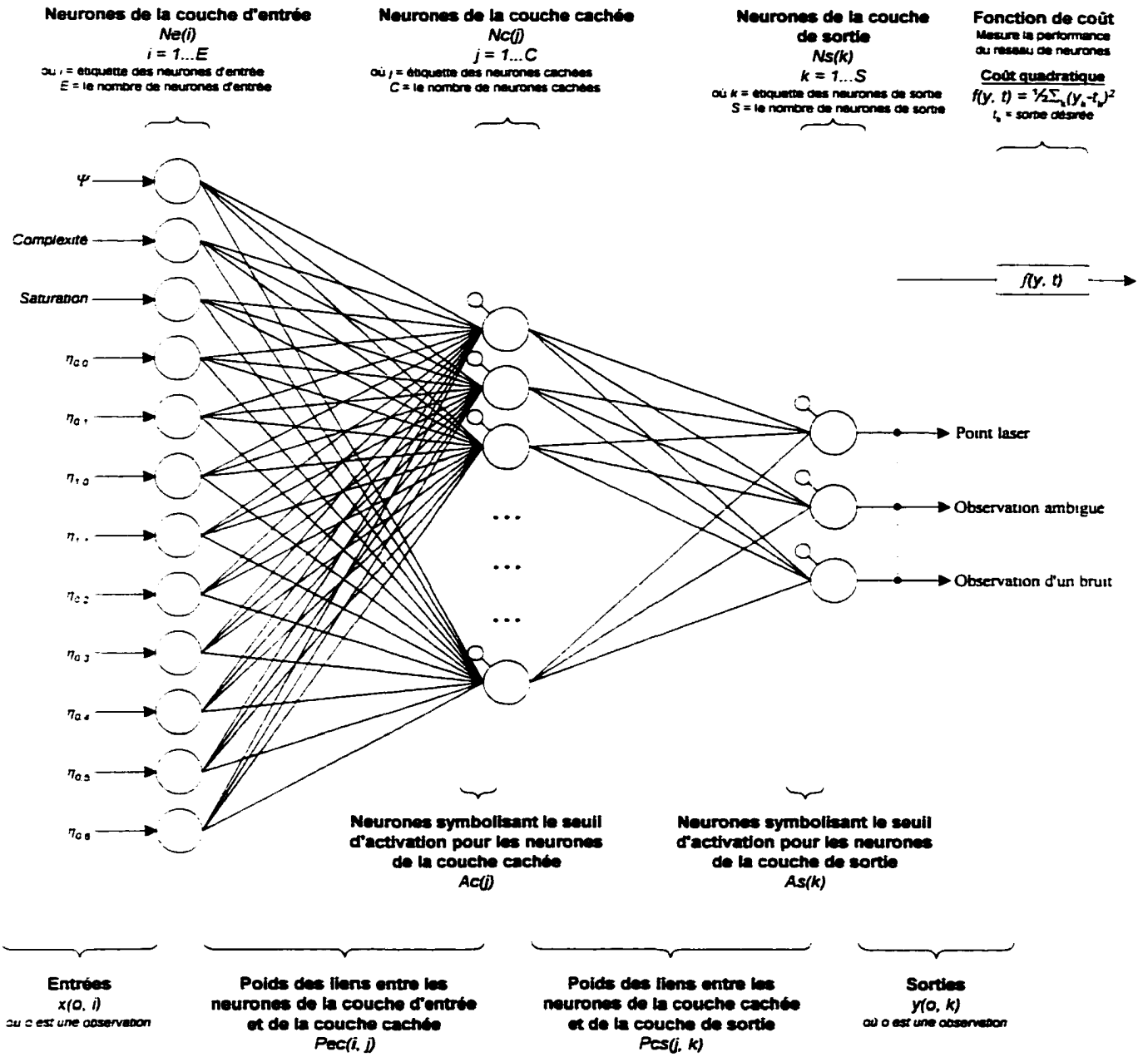


Figure 44 : Architecture du réseau de neurones

La fonction d'activation utilisée est de type tangente hyperbolique. En regardant le profil de la fonction tangente hyperbolique de la figure suivante, on voit que les asymptotes horizontales sont situées à -1 et à 1. Ainsi dans le but de diminuer le nombre

d'époques nécessaires pour que le réseau converge vers une solution, les valeurs des sorties identifiant l'appartenance à une classe sont modifiées pour les intervalles [-1 . -0.8] et [0.8 . 1] au lieu des valeurs 0 et 1.

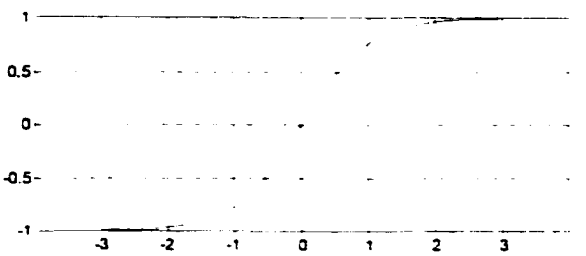


Figure 45 : Profil de la fonction tangente hyperbolique

La définition mathématique du réseau de neurones correspondant à l'architecture présentée et utilisant une fonction d'activation de type tangente hyperbolique est celle-ci :

$$y_k = \tanh\left(As_k + \sum_{j=1}^{C}\left(Pcs_{jk} \cdot \tanh\left(Ac_j + \sum_{i=1}^{E}\left(x_i \cdot Pec_{ij} \right)\right)\right)\right) \tag{4.26}$$

La fonction de coût quadratique globale utilisée se définit ainsi :

$$C = \frac{1}{2}\sum_{o=1}^{O}\sum_{k=1}^{S}\left(y_{ko} - t_{ko}\right)^2 \tag{4.27}$$

Tous les indices et toutes les variables sont définis à même la figure 44.

Pour réaliser l'apprentissage par rétro-propagation du gradient de l'erreur, l'utilisation du diagramme de flot permet de simplifier le problème pour chaque niveau de l'architecture du réseau de neurones. Ce diagramme aide à construire des outils mathématiques simples et puissants faciles à implanter. Le diagramme de flot montré à

la figure suivante montre les quatre constituants du réseau qu'il faut optimiser lors de l'apprentissage (*Pec*, *Ac*, *Pcs* et *As*).

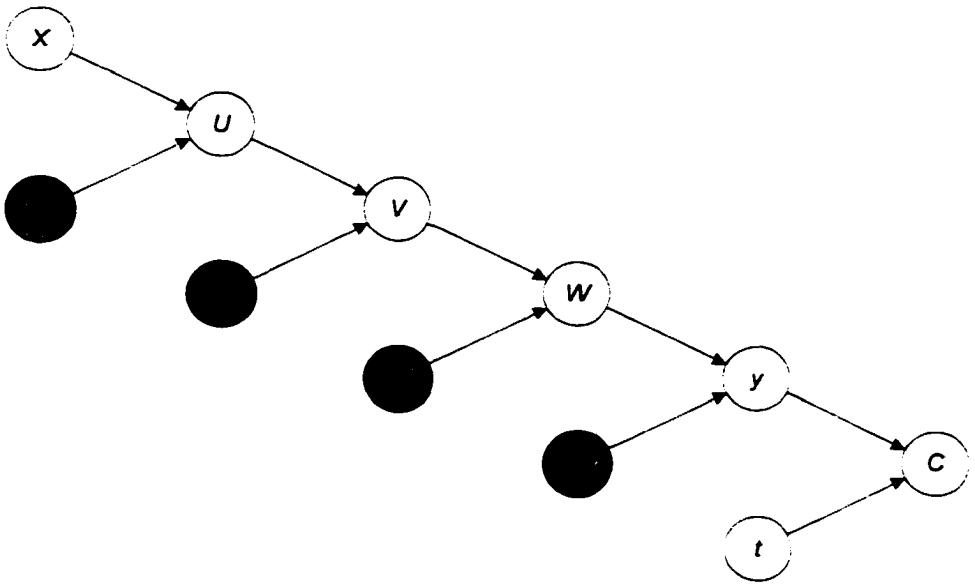


Figure 46 : Diagramme de flot du réseau de neurones

Toutes les équations associées à ce diagramme de flot sont les suivantes :

$$U_j = \sum_{i=1}^{E} Pec_{ij} x_i \tag{4.28}$$

$$V_j = \tanh\left(Ac_j + U_j\right) \tag{4.29}$$

$$W_k = \sum_{j=1}^{C} Pcs_{jk} V_j \tag{4.30}$$

$$y_k = \tanh\left(As_k + W_k\right) \tag{4.31}$$

$$C = \frac{1}{2}\sum_{k=1}^{S}\left(t_k - y_k\right)^2 \tag{4.32}$$

Maintenant, pour calculer le gradient de chaque variable à optimiser, il suffit de calculer la dérivée de chaque paramètre. On remarque la simplicité des calculs issue de la représentation du diagramme de flot. Il suffit d'appliquer la technique de dérivée en chaîne aux équations 4.28 à 4.32. On obtient ainsi :

$$\frac{\partial C}{\partial y_k} = y_k - t_k$$

$$\frac{\partial C}{\partial As_k} = \frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial As_k} = \frac{\partial C}{\partial y_k}\frac{1}{\cosh^2(As_k + W_k)} \tag{4.33}$$

$$\frac{\partial C}{\partial W_k} = \frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial W_k} = \frac{\partial C}{\partial y_k}\frac{1}{\cosh^2(As_k + W_k)} = \frac{\partial C}{\partial As_k}$$

$$\frac{\partial C}{\partial Pcs_{jk}} = \frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial W_k}\frac{\partial W_k}{\partial Pcs_{jk}} = \frac{\partial C}{\partial As_k}V_j \tag{4.34}$$

$$\frac{\partial C}{\partial V_j} = \sum_{k=1}^{S}\frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial W_k}\frac{\partial W_k}{\partial V_j} = \sum_{k=1}^{S}\frac{\partial C}{\partial As_k}Pcs_{jk}$$

$$\frac{\partial C}{\partial Ac_j} = \sum_{k=1}^{S}\frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial W_k}\frac{\partial W_k}{\partial V_j}\frac{\partial V_j}{\partial Ac_j} = \frac{\partial C}{\partial V_j}\frac{1}{\cosh^2(Ac_j + U_j)} \tag{4.35}$$

$$\frac{\partial C}{\partial U_j} = \sum_{k=1}^{S}\frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial W_k}\frac{\partial W_k}{\partial V_j}\frac{\partial V_j}{\partial U_j} = \frac{\partial C}{\partial V_j}\frac{1}{\cosh^2(Ac_j + U_j)} = \frac{\partial C}{\partial Ac_j}$$

$$\frac{\partial C}{\partial Pec_{ij}} = \sum_{k=1}^{S}\frac{\partial C}{\partial y_k}\frac{\partial y_k}{\partial W_k}\frac{\partial W_k}{\partial V_j}\frac{\partial V_j}{\partial U_j}\frac{\partial U_j}{\partial Pec_{ij}} = \frac{\partial C}{\partial Ac_j}x_i \tag{4.36}$$

Tous les outils rendant l'implantation informatique facile sont définis. Ainsi pour chaque exemple d'apprentissage, on calcule d'abord tous les paramètres correspondant à la propagation dans le réseau (U, V, W, y et C) et ensuite tous les paramètres correspondant à la rétro-propagation dans le réseau $\left(\frac{\partial C}{\partial y}, \frac{\partial C}{\partial As}, \frac{\partial C}{\partial Pcs}, \frac{\partial C}{\partial V}, \frac{\partial C}{\partial Ac}\right.$ et $\left.\frac{\partial C}{\partial Pec}\right)$. Les paramètres sont alors modifiés selon l'équation suivante :

$$\theta = \theta - \xi \cdot \frac{\partial C}{\partial \theta} \tag{4.37}$$

où : θ est le paramètre à modifier

ξ est le pas de gradient (aussi appelé constante d'apprentissage)

À la première itération de l'algorithme, il importe de bien initialiser la valeur de chaque paramètre du réseau de neurones. Pour ce faire, les paramètres sont initialisés avec des valeurs tirées d'une distribution uniforme d'amplitude $\left[-\frac{1}{\sqrt{fan-in}}, \frac{1}{\sqrt{fan-in}}\right]$. Le « *fan-in* » est le nombre de paramètres utilisés localement.

Afin d'optimiser la vitesse d'apprentissage du réseau de neurones, il importe de bien ajuster le pas de gradient (ξ). En effet, un pas de gradient trop grand entraîne des oscillations autour d'un minimum de la fonction et empêche cette dernière de converger. Par contre, si le pas de gradient est trop petit, la fonction converge mais en un nombre d'itérations très grand. Le choix d'un bon pas de gradient n'est pas si simple car un bon pas de gradient en début d'apprentissage n'assure pas automatiquement un bon pas à la fin de l'apprentissage. Idéalement, le pas de gradient doit s'adapter en fonction de l'étape d'apprentissage. Il doit être relativement important au début de l'apprentissage et diminuer graduellement tout au long de l'apprentissage. La méthode utilisée ici

correspond au profil d'une courbe qui diminue asymptotiquement vers une valeur minimum. Ainsi après avoir fixé le pas de gradient à la valeur maximum, on le diminue de 0.05 % pour chaque époque. Les valeurs initiale et finale utilisées sont respectivement 0,01 et 0,001, ces valeurs ayant été définies empiriquement. La figure suivante illustre le comportement du pas de gradient :

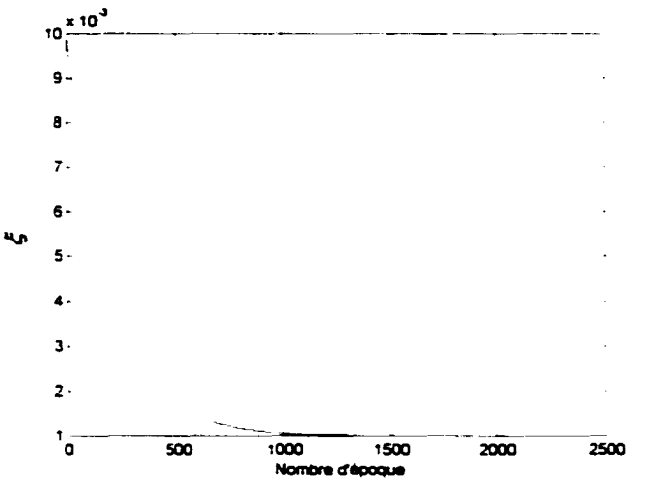


Figure 47 : Évolution de la constante d'apprentissage ξ

Pour réaliser un apprentissage là où le réseau est performant, on doit faire attention à l'apprentissage par cœur. Si l'apprentissage est basé sur un trop grand nombre d'époques, le réseau apprend à classer exactement les observations servant à l'apprentissage. Ainsi le réseau est très performant pour classer des observations identiques à celles de l'ensemble d'apprentissage et est déficient pour classer toute autre observation. C'est comme si la fonction de décision épousait parfaitement les observations de l'ensemble d'apprentissage. Il faut plutôt que les observations servent *d'aimant* à la fonction et que cette dernière s'approche des observations sans les épouser parfaitement. Avec un tel comportement, le réseau de neurones a une capacité de généralisation supérieure. C'est ici qu'intervient le troisième sous-ensemble de données. Pour chaque époque, on fait l'apprentissage à partir du deuxième ensemble de données et on calcule l'erreur de généralisation à partir du troisième ensemble. Tant que l'erreur

de généralisation diminue, le processus continue et le réseau de neurones converge vers un état optimum. Dès que l'erreur de généralisation augmente, on cesse l'apprentissage. Avec cette technique, on évite le sur-apprentissage correspondant à l'apprentissage par cœur.

Le nombre de neurones cachés est directement relié à la capacité du réseau de neurones. Il n'existe pas de méthode analytique permettant de déterminer le nombre optimal de neurones sur la couche cachée. Par contre, il existe une règle empirique :

$$\frac{h}{l} < 1 \tag{4.38}$$

où : h est la capacité du réseau de neurones
l est la taille de l'ensemble d'apprentissage

Puisque $h = 15n_c$ (où n_c est le nombre de neurones sur la couche cachée) et que $l = 1\,800$, on peut dire que $n_c < 120$.

Le nombre de neurones se trouvant sur la couche cachée est le paramètre à optimiser pour ce classificateur. Sachant qu'il ne doit pas y avoir plus de 120 neurones sur cette couche, on fait l'optimisation en faisant varier n_c de 10 à 120 par itération de 10. Pour diminuer le temps requis pour cette optimisation, on limite le nombre d'époques à 2 500. Il est important de bien comprendre qu'ayant limité le nombre d'époques, la performance optimale du réseau de neurones n'est probablement pas atteinte. Mais il importe peu que ces performances soient atteintes puisque seul un ordre de grandeur est nécessaire pour optimiser le paramètre n_c. La figure suivante montre le résultat de cette optimisation.

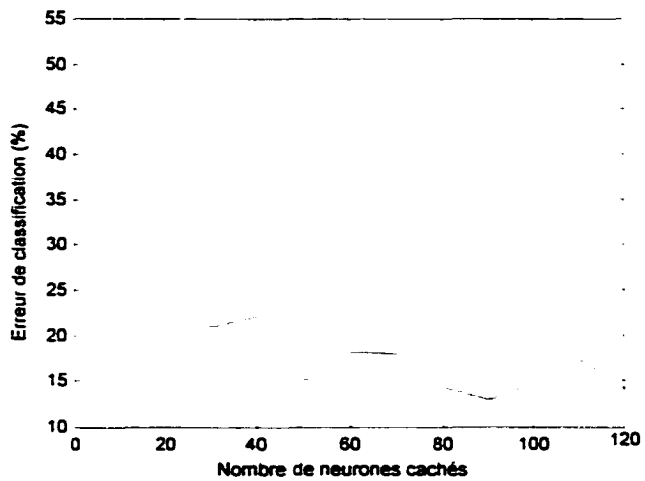


Figure 48 : Optimisation du nombre de neurones sur la couche cachée

C'est avec $n_c = 90$ que le réseau de neurones semble offrir la meilleure performance. La figure suivante illustre l'erreur d'apprentissage (en bas) et l'erreur de généralisation biaisée (en haut).

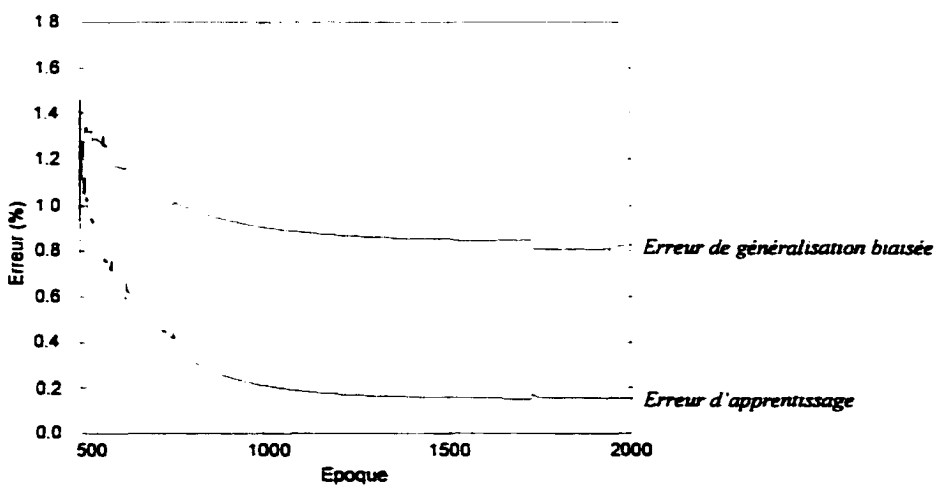


Figure 49 : Erreur d'apprentissage et erreur de généralisation biaisée

Les résultats obtenus de la validation croisée sont résumés dans le tableau suivant.

Tableau VI

Résultats obtenus avec le réseau de neurones

Validation croisée	**Erreur de généralisation non biaisée**		**Temps moyen de classification pour une observation** - ms -
	Erreur totale (sur les 3 classes) - % -	***Erreur sur la classe des points laser*** - % -	
Ensemble test 1	14,50	1,00	0,0230
Ensemble test 2	14,83	0,50	0,0260
Ensemble test 3	13,50	0,67	0,0251
Ensemble test 4	13,83	1,17	0,0251
Ensemble test 5	13,17	1,33	0,0261
Ensemble test 6	14,17	1,00	0,0260
Ensemble test 7	14,00	0,83	0,0275
Ensemble test 8	14,50	1,17	0,0250
Ensemble test 9	13,83	0,83	0,0261
Ensemble test 10	14,50	1,17	0,0250
Moyenne	14,08	0,97	0,0255
Écart-type	0,52	0,26	0,0012

Les résultats obtenus du réseau de neurones sont très satisfaisants. Un taux d'erreur inférieur à 1 % pour la classification des points laser indique que la performance est excellente pour des images prises dans des conditions moins complexes. De plus, le temps de calcul requis pour une classification est minime. On voit qu'il est possible de classifier près de 40 000 observations par seconde.

4.2.5 Comparaison entre les trois classificateurs

Le tableau suivant montre les résultats obtenus pour les trois classificateurs.

Tableau VII

Comparaison des résultats entre les trois classificateurs

	Bayes quadratique	***K-NN***	**Réseau de neurones**
Erreur de classification pour la classe de points laser - % -	5,40 ± 0,56	0,18 ± 0,20	0,97 ± 0,26
Temps de calcul pour une observation - ms -	2,1713 ± 0,0671	141,700 ± 1,098	0,0255 ± 0,0012

Il est évident que le classificateur retenu est le réseau de neurones. Premièrement, il offre des performances de classification très près du *K-NN* et de beaucoup supérieures au classificateur Bayes. Deuxièmement, le temps de calcul est beaucoup moindre que les deux autres, soit 85 fois plus rapide que le classificateur Bayes et 5 557 fois plus rapide que le *K-NN*. Il ne faut pas perdre de vue que les performances du temps de calcul pour le classificateur *K-NN* peuvent être augmentées en réduisant l'ensemble de données.

Plusieurs tests ont montrés que les performances de classification du réseau de neurones avec des images de complexités *normales* sont de 100 %.

L'erreur de classification pour les points laser n'indique pas le nombre de points laser se trouvant sur la scène. Elle indique plutôt en terme de pourcentage le nombre d'observations classées comme étant des points laser alors qu'elles sont du bruit ou des observations ambiguës. Ces erreurs de classification peuvent engendrer de fausses insertions dans la représentation virtuelle de l'environnement et éventuellement une mauvaise planification de trajectoire.

4.3 Processus algorithmique

Tous les éléments de résolution du problème d'implantation algorithmique sont maintenant identifiés. Le processus algorithmique réalisé est constitué des six étapes illustrées sur la figure 34. Les prochaines sections montrent de façon détaillée toutes les étapes du processus.

4.3.1 Calibrage des caméras

Le calibrage des caméras permet de trouver la relation entre les coordonnées spatiales de la scène et les coordonnées images. On utilise la technique présentée à la section 2.2. Une cible constituée d'une matrice de points disposés sur trois plans sert d'image de référence.

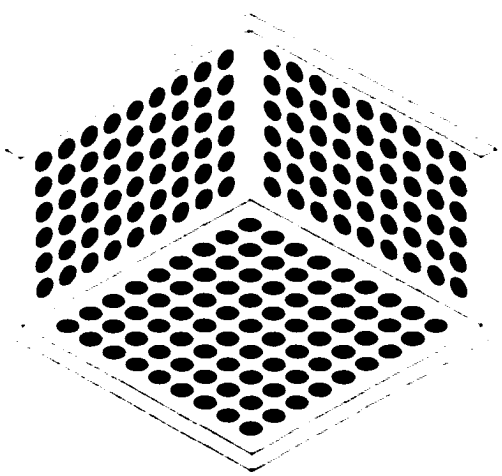


Figure 50 : Grille de calibrage

La position précise du centre des points est connue par rapport au référentiel de l'environnement. On détermine le centroïde des points laser par rapport au référentiel image par un simple traitement d'image. En se référant à l'équation 2.5, chaque point de l'espace donne une coordonnée de l'environnement (x_e, y_e, z_e) et chaque point de l'image donne une coordonnée image (x_i, y_i). On évalue ainsi la matrice de projection perspective par l'équation 2.6.

Il existe plusieurs autres techniques pour réaliser le calibrage des caméras. Certaines sont incontournables comme la méthode de Tsai (1986). Cette méthode est d'autant plus répandue et simplifiée qu'il existe une bibliothèque en langage C disponible sur Internet[19] permettant de faire tous les traitements nécessaires au calibrage. De plus, le modèle utilisé par cette technique tient compte de la distorsion radiale.

Voisin, Marzani et Diou (2000) proposent une technique de calibrage utilisant un plan mobile sur lequel est projetée une matrice de points laser. En relevant plusieurs plans images dont chaque position est connue, on détermine les équations des droites correspondant aux faisceaux laser et permettant d'évaluer directement les paramètres de la caméra. Cette technique est tout à fait indiquée pour le système de vision développé puisqu'elle utilise directement le projecteur laser sans aucune cible de référence. En fait, d'un point de vue pratique, cette technique peut permettre au robot de faire un auto-calibrage du système de vision simplement en s'approchant ou en s'éloignant d'une surface connue.

4.3.2 Acquisition des images

L'acquisition des images est certainement l'étape la plus délicate et la plus importante du processus. Étant donné que tout le traitement des données est basé sur l'analyse des images, il importe que ces dernières représentent bien la réalité et offrent un maximum de qualité. Malgré le fait qu'aucun algorithme ne soit nécessaire pour réaliser cette partie du processus, cette section présente sous forme de réflexions quelques points importants reliés à l'acquisition des images.

---

[19] Voir : www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html

4.3.2.1 Synchronisation des caméras

Tout système de perception stéréoscopique doit acquérir l'information de la scène observée exactement au même moment pour tous les capteurs impliqués. On peut illustrer ce problème en prenant le cas où il y a une différence de temps entre la prise de l'image de la caméra de gauche et de la caméra de droite. Ce délai engendre inévitablement une différence de position et d'orientation des caméras entre les deux prises d'image. De plus, il est possible que les acteurs de la scène se soient déplacés. Ainsi il est impossible d'évaluer avec précision la position spatiale réelle des acteurs de la scène. C'est pourquoi il est important de synchroniser la prise d'informations.

Plusieurs caméras disponibles sur le marché offrent la possibilité de synchroniser la prise d'images de N caméras selon un signal précis.

4.3.2.2 Éclairage ambiant – puissance du laser – sensibilité des caméras

On voit à la section 1.3.6 qu'il existe une double contrainte contradictoire pour déterminer la puissance du laser : elle doit être la plus forte possible pour contrer la lumière ambiante et la plus faible possible pour augmenter la sécurité reliée à son utilisation. En considérant un laser de puissance ajustable, des caméras avec une sensibilité ajustable et des photodétecteurs pour déterminer le niveau d'éclairage ambiant, il est possible de maximiser l'utilisation de la puissance laser tout en minimisant l'impact relié à la sécurité.

Lorsque la lumière ambiante est très faible, il n'est pas nécessaire d'avoir un laser puissant. En augmentant le gain en sensibilité des caméras et en diminuant la puissance du laser, il est possible d'avoir des images très nettes et de minimiser l'impact néfaste des faisceaux laser. Il ne faut pas perdre de vue que l'œil humain adapte aussi sa sensibilité en fonction de l'éclairage ambiant (en ouvrant et en fermant sa pupille par exemple). Si les caméras imitent le comportement de l'œil, il faut diminuer la puissance des lasers pour éviter de saturer les images prises.

De la même façon, si le système se retrouve dans un endroit où l'éclairage ambiant est très fort, le gain des caméras peut être diminué (imitant le comportement des yeux) et par le fait même la puissance des lasers peut être augmentée. Encore une fois, il est possible d'avoir des images nettes et la sécurité associée au laser est ainsi optimisée.

Ces méthodes ne sont pas implantées sur le prototype présenté dans ce mémoire. Par contre, étant donné la simplicité d'un tel développement, il serait très sage que ces ajouts fassent partie des premières améliorations à considérer lors des développements futurs. En fait, une simple expérimentation permettrait de déterminer empiriquement la relation existant entre l'intensité de la lumière ambiante, la puissance du laser et la sensibilité des caméras.

D'un autre côté, il ne faut pas penser que ces techniques suffisent à rendre sécuritaire l'utilisation de n'importe quel laser. Tel qu'il est montré à la section 3.1, il faut soigneusement choisir la puissance maximale du laser pour qu'il respecte les normes de sécurité en vigueur.

4.3.3 Extraction des observations

Cette étape du processus algorithmique est celle qui demande le plus de puissance de calcul. Elle consiste à trouver les points laser sur les images.

La première étape consiste à déterminer de façon automatique le seuil de segmentation de l'image. La deuxième étape est de segmenter l'image et de localiser toutes les observations pouvant potentiellement être des points laser sur les images. Étant donné que les images peuvent avoir un certain niveau de bruit, il est essentiel de classer les observations trouvées afin d'identifier les points laser et le bruit. Pour réaliser cette étape, il faut calculer la valeur numérique des descripteurs de formes décrivant chacune des observations.

4.3.3.1 Détermination automatique du seuil de segmentation

La définition automatique d'un seuil de segmentation est une tâche très délicate. La complexité des images demande parfois l'utilisation de techniques complexes. Par contre, la simplicité des images obtenues facilite le travail et permet le développement d'un algorithme robuste et performant. La raison est fort simple : l'usage du laser et des filtres interférentiels donne des images dont les histogrammes de luminance sont bimodales, c'est-à-dire avec deux régions distinctes facilement identifiables.

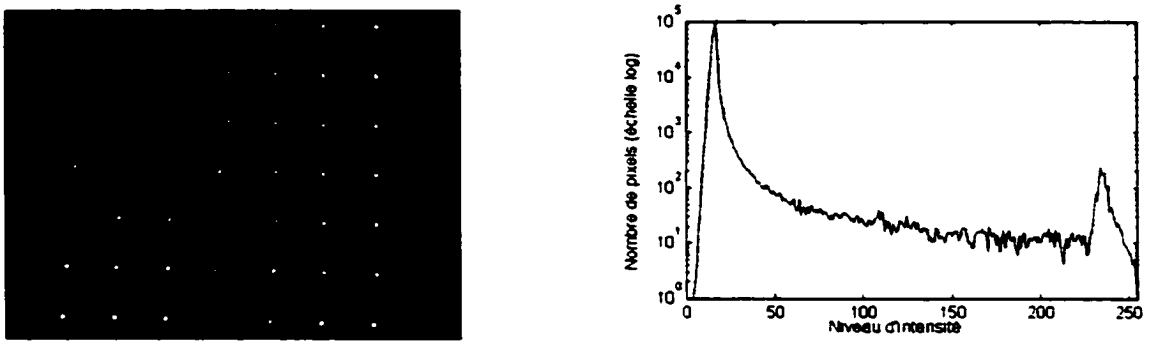


Figure 51 : Exemple d'histogramme bimodale obtenu (échelle logarithmique)

Ainsi un simple algorithme de recherche de seuil basé sur les propriétés des histogrammes bimodales permet de segmenter l'image de façon adéquate.

En supposant que tous les modes de l'histogramme correspondent à une distribution gaussienne, ces modes peuvent être décrits mathématiquement ainsi :

$$f_f(x) = \frac{1}{\sigma_f \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu_f}{\sigma_f}\right)^2} \text{ et } f_s(x) = \frac{1}{\sigma_s \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu_s}{\sigma_s}\right)^2} \tag{4.39}$$

où : μ_f et σ_f représentent la moyenne et l'écart type de la distribution correspondant au fond

μ_s et σ_s représentent la moyenne et l'écart type de la distribution correspondant au signal

Le seuil optimal est l'intersection des deux courbes, c'est-à-dire lorsque $f_f(x) = f_s(x)$. Ainsi on montre que le seuil de segmentation optimal est défini par l'équation suivante :

$$S = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a} \tag{4.40}$$

où : S est le seuil de segmentation

$$a = \sigma_s^2 - \sigma_f^2$$

$$b = 2 \cdot \left(\sigma_f^2 \cdot \mu_s - \sigma_s^2 \cdot \mu_f \right)$$

$$c = \sigma_s^2 \cdot \mu_f^2 - \sigma_f^2 \cdot \mu_s^2 - 2 \cdot \sigma_f^2 \cdot \sigma_s^2 \cdot \ln\left(\frac{\sigma_s}{\sigma_f} \right)$$

Il suffit seulement de localiser la moyenne de chaque courbe gaussienne pour ensuite calculer les écarts types associés et finalement la position optimale du seuil de segmentation. De l'équation 4.40, on obtient deux racines. Il suffit de prendre la racine contenue entre les deux moyennes (c'est-à-dire $s \in [\mu_1, \mu_2]$).

4.3.3.2 Segmentation et localisation des observations

L'algorithme utilisé pour localiser chacune des observations est fort simple. En partant du coin supérieur gauche de l'image, on parcourt l'image de gauche à droite et de haut en bas. On recherche un premier pixel possédant une intensité supérieure au seuil de segmentation préalablement déterminé. Lorsqu'un tel pixel est trouvé, on applique un algorithme de recherche de contour permettant de localiser tous les éléments qui forment les frontières de l'observation.

Parcourir tous les pixels de l'image pour localiser les observations est un processus très long et surtout inutile. Les images sont de 640 × 480 pixels et possèdent ainsi 307 200 pixels. On peut réduire la résolution de l'image au quart de sa dimension (c'est-à-dire au seizième de sa surface) pour limiter la recherche sans perte d'informations. Ainsi on a des images de 160 × 120 pixels qui totalisent 19 200 pixels (un gain de 93,75 % en terme de surface à parcourir et de temps de calcul). La figure suivante illustre la proportion que prend l'image réduite par rapport à l'image originale.

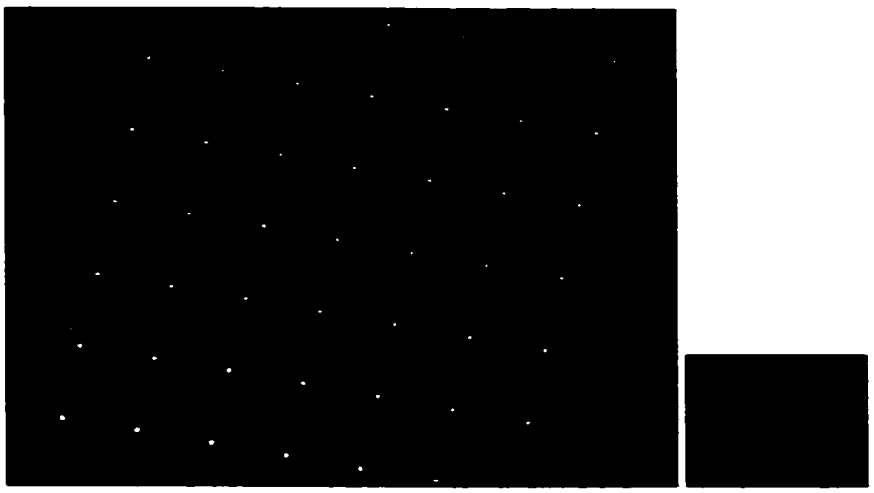


Figure 52 : Image multirésolution

Lorsqu'une observation est identifiée sur l'image à faible résolution, on poursuit l'algorithme de recherche et d'identification sur l'image originale.

4.3.3.3 Calcul des attributs

Après avoir localisé les observations sur une image, il faut calculer les valeurs numériques des attributs servant au classificateur. On estime qu'après la localisation des observations telle que décrite à la section précédente, les attributs suivants sont déjà évalués : le contour défini par le code de Freeman, le nombre de pixels et le périmètre. Les deux derniers attributs sont calculés à l'aide du code de Freeman.

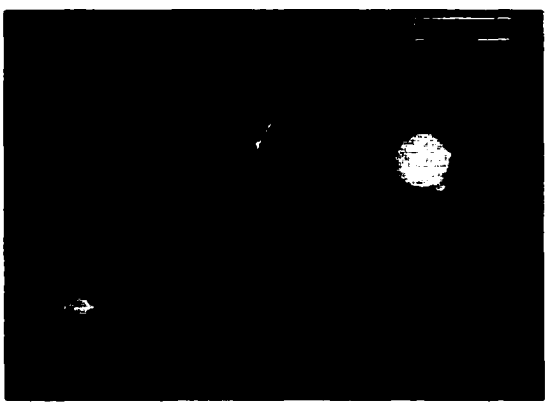
Pour chaque observation ayant le nombre de pixels compris dans l'intervalle [9, 125] (voir la section 4.1.1.2), on calcule tous les attributs requis.

L'algorithme proposé est très simple : il consiste à balayer trois fois la surface englobant l'observation pour calculer chacun des attributs. Voici ce que chacun des balayages permet de calculer :

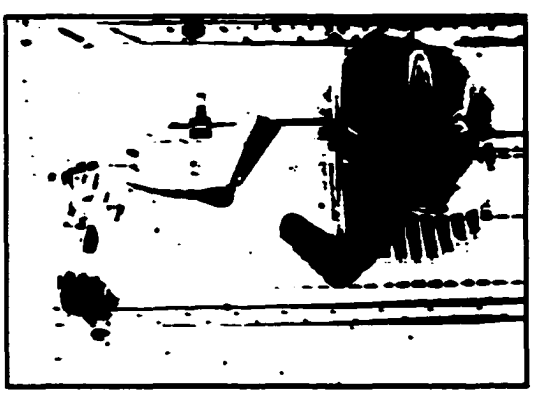
1. les moments d'inertie, ce qui donne l'aire, la complexité, le centroïde et la saturation;

2. les paramètre α et $\bar{I}$ servant à l'estimation de Ψ et les moments d'inertie centrés. ce qui donne les moments de Pen et Keane;
3. Ψ.

Les images de la figure suivante montrent graphiquement les étapes de recherche et d'identification des observations à partir d'une image très bruitée.



a) Image acquise



b) Image segmentée

c) Identification des observations

Figure 53 : Processus algorithmique de recherche et de localisation des observations

4.3.4 Classification des observations

Tel que décrit à la section 4.2.5, le réseau de neurones de type perceptron multicouche est le classificateur sélectionné pour l'application et offre des performances remarquables autant en termes de temps de calcul que d'erreur de classifications. La

section 4.2.4 présente les différents outils mathématiques permettant d'implanter ce type de solution.

Il suffit de mettre à l'entrée du réseau de neurones la valeur des attributs précédemment calculée et ensuite de faire la propagation dans le réseau. Pour que cette observation soit considérée comme étant un point laser, la sortie du réseau doit respecter deux contraintes : la sortie associée à la classe des points laser doit être plus grande que zéro et cette sortie doit être exclusive, c'est-à-dire que les autres sorties doivent être inférieures à zéro.

4.3.5 Appariement

Il existe plusieurs méthodes algorithmiques permettant de solutionner le problème d'appariement. Par exemple, Horaud et Monga (1993) présentent quatre méthodes très intéressantes : l'appariement par mise en correspondance hiérarchique, l'appariement par programmation dynamique (basé sur l'algorithme de Viterbi), l'appariement par relaxation et l'appariement par isomorphisme de graphe. D'autres traitent de ces approches. Falkenhagen (1995) a implanté une méthode basée sur la programmation dynamique et montre comment modifier l'algorithme de Viterbi pour solutionner le problème. Ayache (1989) explique en détail toutes les étapes de l'algorithme d'appariement par isomorphisme de graphe.

Geiger. Ladendorf et Yuille (1994) présentent une solution élégante aux problèmes d'occlusion basée sur une approche bayesienne tirant profit des phénomènes d'occlusion. Ils utilisent aussi une méthode de programmation dynamique pour solutionner le problème du meilleur appariement. Une nouvelle approche est introduite par Tomasi et Manduchi (1996). Cette approche, basée sur les courbes intrinsèques, permet de voir et de solutionner différemment le problème de stéréo-correspondance. Trucco et Verri (1998) présentent pour leur part, une solution très simple en calculant la distance dans l'espace des caractéristiques entre une observation d'une image et toutes

les autres observations de celle-ci. Cette distance sert de critère de similitude et permet de localiser les meilleurs appariements.

L'algorithme implanté correspond à une méthode réduite de l'appariement par programmation dynamique. La simplification faite découle du peu d'éléments des images prises. En fait, ces images présentent si peu d'observations et ces dernières sont généralement si bien distribuées dans l'image qu'on ne retrouve que quelques observations situées sur les droites épipolaires conjuguées. Toutes les paires d'images étudiées ne présentent jamais plus de deux groupes de quatre points sur les mêmes droites épipolaires conjuguées, même sur les images les plus complexes. Il est évident qu'il reste possible que le cas limite où toutes les observations sont alignées le long des droites épipolaires puisse arriver. Dans un tel cas, l'algorithme implanté est moins efficace.

L'idée est simple : pour chaque groupe d'observations se retrouvant sur les droites épipolaires conjuguées, on calcule une matrice de coût (C) pour laquelle chaque élément correspond à un appariement possible. Cette matrice est de dimension $n_g \times n_d$, n_g étant le nombre d'observations sur l'image de gauche et n_d étant le nombre d'observations sur l'image de droite. De cette matrice, on crée une liste de tous les ensembles possibles d'appariement (L). Si on omet la contrainte d'unicité, le nombre d'ensembles possibles devient rapidement énorme. Avec la contrainte d'unicité, il est possible de réduire la taille de L considérablement. Ensuite on calcule le coût total de chaque appariement et on choisit finalement le moins coûteux.

Cet algorithme est simple et rapide pour de petits ensembles. Lorsque C atteint une certaine taille, l'algorithme devient largement déficient et ne permet pas de converger rapidement vers la solution. L'algorithme de Viterbi permet de trouver très rapidement l'ensemble d'appariements minimisant les coûts.

On calcule le coût d'un appariement par une somme pondérée des différentes contraintes implantées. On normalise les valeurs des contraintes calculées puisque les unités sur les calculs sont toutes d'amplitude différente.

$$C_{ij} = \alpha_1 \tilde{e}_{ij} + \alpha_2 \tilde{d}_{ij} + \alpha_3 \tilde{c}_{ij} \tag{4.41}$$

où : C_{ij} est l'élément de la matrice de coût correspondant à la $i^{ième}$ observation sur l'image de gauche et la $j^{ième}$ observation sur l'image de droite. Ces observations se trouvent dans l'intervalle défini des droites épipolaires conjuguées.

α_1, α_2 et α_3 sont les coefficients de pondération

$\tilde{e}_{ij}$ est l'indice normalisé de la contrainte épipolaire

$\tilde{d}_{ij}$ est un indice normalisé de la contrainte de la limite de disparité

$\tilde{c}_{ij}$ est un indice normalisé de la contrainte de corrélation

La contrainte épipolaire est déjà utilisée pour limiter le nombre d'appariements lors de la création de L puisqu'on ne considère que les appariements potentiels des observations se trouvant sur les droites épipolaires conjuguées. Cette contrainte épipolaire peut être utilisée d'avantage. En effet, même si un intervalle défini permet d'éliminer systématiquement les candidats indésirables, il reste que les candidats restants n'ont pas tous le même poids. Ceux se trouvant près des limites permises sont moins intéressants que ceux ayant l'ordonnée identique à l'observation. De plus, on peut affirmer que la distance entre les ordonnées génère un coût non linéaire mais plutôt exponentiel. On fixe arbitrairement le coût proportionnel au carré de la distance des ordonnées.

$$\tilde{e}_{ij} = \left(\frac{y_i^r - y_j^r}{2\Delta y} \right)^2 \tag{4.42}$$

où : $\tilde{e}_{ij}$ est l'indice normalisé de la contrainte épipolaire

y_h^r est l'ordonnée rectifié de la $h^{ième}$ observation

Δy est la distance maximale tolérée pour qu'une observation soit considérée sur la droite épipolaire conjuguée

La contrainte de la limite de disparité est intégrée par une fonction échelon. Les observations se trouvant à l'intérieur de l'intervalle de disparité sont retenues et les autres rejetées.

$$\tilde{d}_{ij} = \begin{cases} 0 & \text{si } k^r_{\min_i} < x^r_j < k^r_{\max_i} \\ 1 & \text{sinon} \end{cases} \tag{4.43}$$

où : $\tilde{d}_{ij}$ est l'indice normalisé de la contrainte épipolaire

x^r_h est l'abscisse rectifiée de la $h^{ième}$ observation

$k^r_{\min_h}$ et $k^r_{\max_h}$ sont les abscisses rectifiées des bornes de l'intervalle de disparité calculées selon les équations 2.14 et 2.15 pour la $h^{ième}$ observation

Maintenant on considère l'indice de corrélation de l'intensité entre les deux régions des images tel que présenté à la section 2.4.9. L'indice de corrélation normalisé se calcule ainsi :

$$\tilde{c}_{ij} = \frac{\sum_{k=-M}^{M}\sum_{l=-N}^{N}\left(I^g(x_i-k, y_i-l) - I^d(x_j-k, y_j-l)\right)^2}{(2M+1)\cdot(2N+1)\cdot I_{\max}} \tag{4.44}$$

où : $\tilde{c}_{ij}$ est l'indice normalisé de la corrélation entre deux régions

x_h et y_h sont les coordonnées de la $h^{ième}$ observation

I^g et I^d sont les images gauche et droite

I_{max} correspond à la résolution en intensité pour chaque pixel (ici on a $2^8 - 1 = 255$)

$(2M + 1) \times (2N + 1)$ est la taille de la région analysée et est définie au double de la taille de l'observation

Maintenant, les facteurs de pondération sont définis empiriquement :

- $\alpha_1 = 0{,}6$
- $\alpha_2 = \infty$ (la valeur infinie est imposée parce que le respect de la contrainte de la limite de disparité est obligatoire)
- $\alpha_3 = 1 - \alpha_1 = 0{,}4$

Après avoir calculé les coûts pour chaque appariement possible, il faut identifier l'ensemble de toutes les combinaisons d'appariements possibles (L) avec les observations relevées. Puisqu'il est possible que certaines observations ne puissent être appariées entre elles, il faut définir un coût pour les observations non appariées. Ce coût représente en fait une pénalité imposée pour favoriser le maximum d'appariements possibles. Ainsi le coût total est :

$$A_i = \sum_{\forall i \in L_i} C + \eta_i \cdot \Phi \tag{4.45}$$

où : $\sum_{\forall i \in L_i} C$ représente le coût cumulé de la $i^{ième}$ combinaison d'appariement possible de L

η représente le nombre d'observations qui peuvent être appariées et qui ne le sont pas.

$$\eta \in \left[0 \quad \min(n_g, n_d)\right]$$

Φ est le coût de non appariement. Il est défini empiriquement égal à 0,3

Le tableau suivant est un exemple illustrant toutes les étapes du processus algorithmique pour l'appariement.

Tableau VIII

Exemple d'appariement

Étape 1

Acquisition des images

Image de gauche

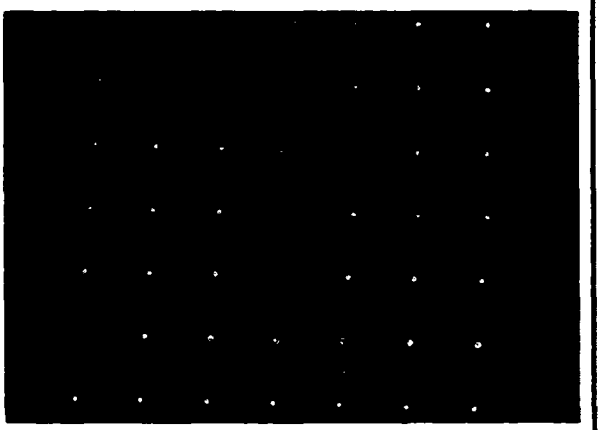
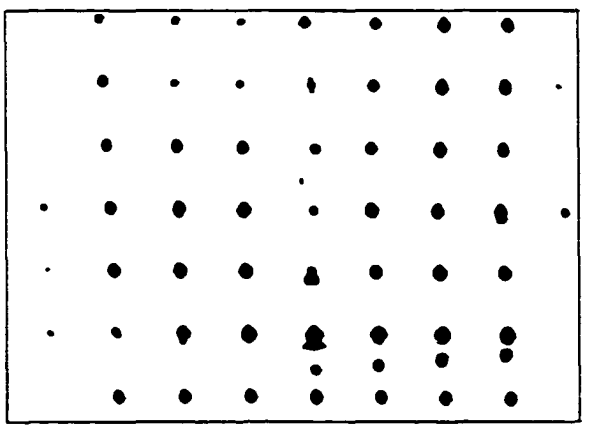


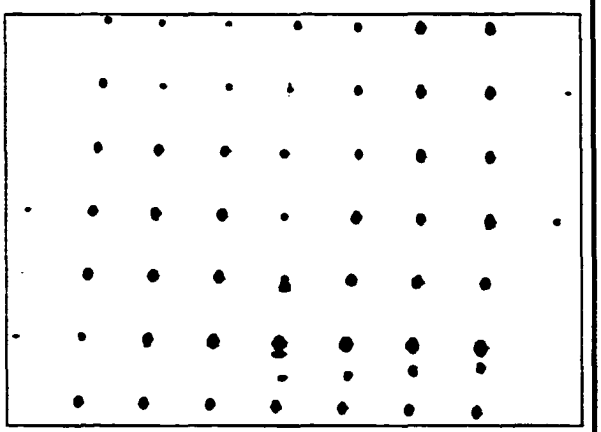
Image de droite

Étape 2

Extraction des observations



Observations de l'image de gauche



Observations de l'image de droite

Tableau VIII (suite)

Étape 3

Rectification

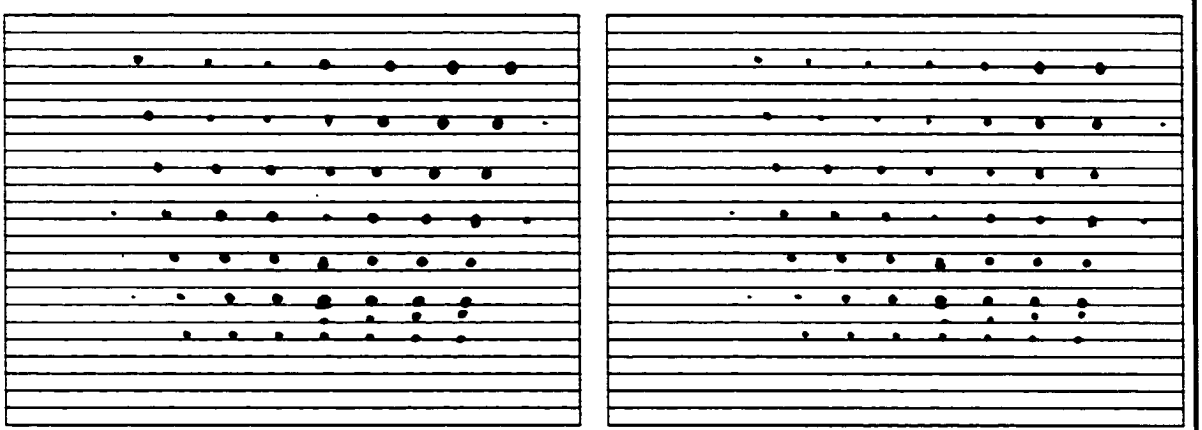


Image de gauche rectifiée

Image de droite rectifiée

- Ces images représentent l'étape de rectification. En fait, seules les coordonnées des observations relevées à l'étape 2 sont rectifiées.
- Les lignes horizontales représentent les droites épipolaires rectifiées.

Étape 4

Sélection des observations se trouvant sur les mêmes droites épipolaires conjuguées

Observations de l'image de gauche retenues pour un ensemble d'appariement

Observations de l'image de droite retenues pour un ensemble d'appariement

- Ces observations se trouvent à l'intérieur de l'intervalle défini $\pm\Delta y$ autour des droites épipolaires conjuguées.
- Les trois observations illustrées ici sont celles se trouvant dans le coin inférieur droit des images.

Tableau VIII (suite)

Étape 5

Calcul de la matrice de coûts

$$\tilde{e} = \begin{array}{c|ccc} & A1 & A2 & A3 \\ B1 & 0.0021 & 0.0118 & 0.0647 \\ B2 & 0.0324 & 0.0007 & 0.0144 \\ B3 & 0.1286 & 0.0417 & 0.0034 \end{array}$$

$$\tilde{d} = \begin{array}{c|ccc} & A1 & A2 & A3 \\ B1 & 0 & 0 & 0 \\ B2 & 0 & 0 & 0 \\ B3 & 0 & 0 & 0 \end{array}$$

$$\tilde{c} = \begin{array}{c|ccc} & A1 & A2 & A3 \\ B1 & 0.0198 & 0.0364 & 0.0343 \\ B2 & 0.0395 & 0.0187 & 0.0291 \\ B3 & 0.0375 & 0.0312 & 0.0125 \end{array}$$

$$C = \alpha_1\tilde{e} + \alpha_2\tilde{d} + \alpha_3\tilde{c} = \begin{array}{c|ccc} & A1 & A2 & A3 \\ B1 & 0.0092 & 0.0216 & 0.0525 \\ B2 & 0.0353 & 0.0079 & 0.0203 \\ B3 & 0.0921 & 0.0375 & 0.0071 \end{array}$$

Tableau VIII (suite)

Étape 6 *Identification des appariements possibles et de leur coût*				
Ensembles d'appariements possibles	***Liste des éléments de ces ensembles***	***Ensembles d'appariements possibles***	***Liste des éléments de ces ensembles***	***Coût total de ces ensembles***
Sans appariement 1 ens. possible $\eta = 3$	∅	**Sans appariement** 1 ens. possible $\eta = 3$	∅	0,900
Avec un seul appariement 9 ensembles possibles $\eta = 2$	A1B1 A2B1 A3B1 A1B2 A2B2 A3B2 A1B3 A2B3 A3B3	**Avec un seul appariement** 9 ensembles possibles $\eta = 2$	A1B1 A2B1 A3B1 A1B2 A2B2 A3B2 A1B3 A2B3 A3B3	0,6092 0,6216 0,6525 0,6353 0,6079 0,6203 0,6921 0,6375 0,6071
Combinaisons de deux appariements 36 ensembles possibles $\eta = 1$	A1B1 - A1B2 A1B1 - A1B3 A1B1 - A2B1 A1B1 - A2B2 A1B1 - A2B3 A1B1 - A3B1 A1B1 - A3B2 A1B1 - A3B3 A1B2 - A1B3 A1B2 - A2B1 A1B2 - A2B2 A1B2 - A2B3 A1B2 - A3B1 A1B2 - A3B2 A1B2 - A3B3 A1B3 - A2B1 A1B3 - A2B2 A1B3 - A2B3 A1B3 - A3B1 A1B3 - A3B2 A2B1 - A2B2 A2B1 - A2B3 A2B1 - A3B1 A2B1 - A3B2 A2B1 - A3B3 A2B2 - A2B3 A2B2 - A3B1 A2B2 - A3B2 A2B3 - A3B1 A2B3 - A3B2 A2B3 - A3B3	**Combinaisons de deux appariements** 18 ensembles possibles $\eta = 1$	A1B1 - A2B2 A1B1 - A2B3 A1B1 - A3B2 A1B1 - A3B3 A1B2 - A2B1 A1B2 - A2B3 A1B2 - A3B1 A1B2 - A3B3 A1B3 - A2B1 A1B3 - A2B2 A1B3 - A3B1 A1B3 - A3B2 A2B1 - A3B2 A2B1 - A3B3 A2B2 - A3B1 A2B2 - A3B3 A2B3 - A3B1 A2B3 - A3B2	0,3171 0,3467 0,3295 0,3163 0,3569 0,3728 0,3878 0,3424 0,4137 0,4000 0,4446 0,4124 0,3419 0,3287 0,3604 0,3150 0,3900 0,3578
Combinaisons de trois appariements 84 ens. possibles $\eta = 0$	... A1B1 - A2B1 - A3B1 A1B1 - A2B1 - A3B2 A1B1 - A2B1 - A3B3 A1B1 - A2B2 - A3B1 ...	**Combinaisons de trois appariements** 6 ens. possibles $\eta = 0$	A1B1 - A2B2 - A3B3 A1B1 - A2B3 - A3B2 A1B2 - A2B1 - A3B3 A1B2 - A2B3 - A3B1 A1B3 - A2B1 - A3B2 A1B3 - A2B2 - A3B1	0,0242 0,0670 0,0640 0,1253 0,1340 0,1525

Tableau VIII (Suite)

Étape 7 *Sélection du meilleur ensemble d'appariements*
On sélectionne l'ensemble d'appariement qui présente le coût le plus faible. ***A1B1 - A2B2 - A3B3***

4.3.6 Reconstruction 3D des coordonnées spatiales

La reconstruction 3D est la dernière étape du processus algorithmique du capteur stéréoscopique. Elle permet de générer une liste de points 3D représentant une partie de l'environnement.

Pour chaque appariement trouvé (x^d, y^d) - (x^g, y^g), on utilise la matrice de projection perspective pour calculer à l'aide de l'équation 2.19 la coordonnée de la scène correspondante.

L'insertion des points dans la carte virtuelle est présentée dans la deuxième partie du mémoire.

CHAPITRE 5

PRÉSENTATION DES RÉSULTATS

Au moment de rédiger ce mémoire, aucun véhicule mobile (sur roues ou sur pattes) ne permet de faire des tests en mouvement car les prototypes ne sont pas encore prêts. Ainsi le système de vision est testé et validé à l'aide d'un montage statique. Le prototype de vision est monté sur un trépied selon la configuration nominale sur le robot (pour la hauteur et l'orientation – voir la section 3.4).

5.1 Scène modélisée

La figure suivante montre une scène modélisée permettant l'évaluation des performances globales du capteur. Le rôle premier de cette expérimentation est de valider et de vérifier la fonctionnalité globale du matériel et du logiciel. Ensuite, quelques indices de performances permettent d'évaluer la performance globale du capteur. Ces indices sont :

a. erreur de classification des points laser;

b. erreur d'appariement;

c. précision sur les mesures;

d. fréquence d'échantillonnage.

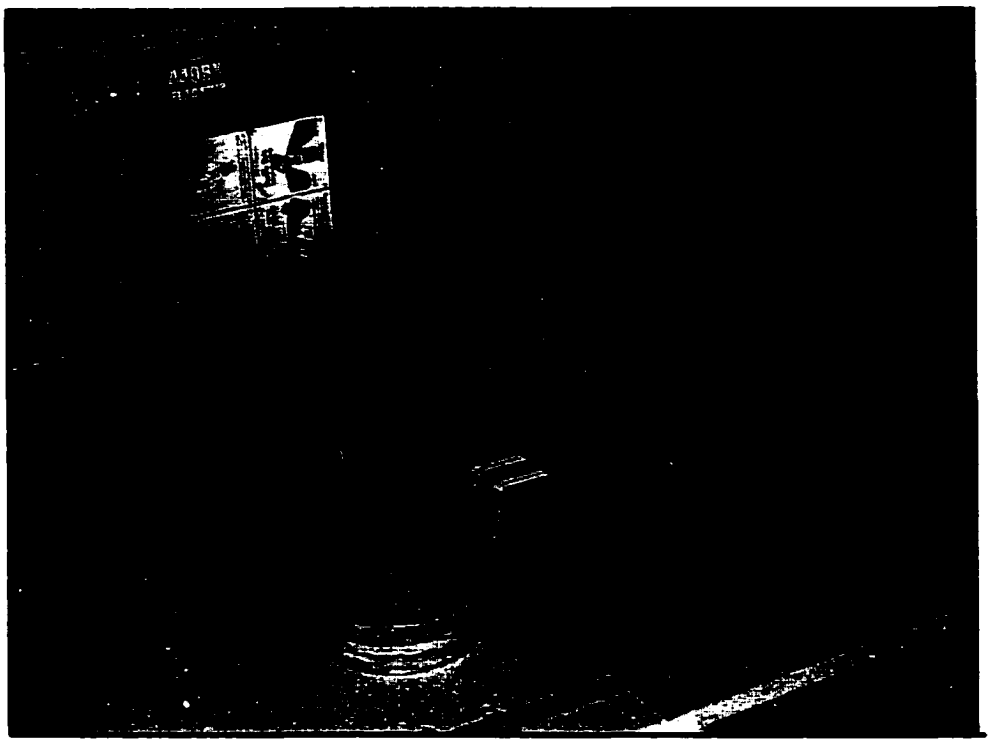


Figure 54 : Scène modélisée

Cette scène est constituée d'un panneau de polystyrène sur lequel est déposé cinq boîtes de dimensions différentes. On retrouve trois boîtes standard en carton, une boîte cylindrique en métal et une boîte en plastique noir peu réfléchissant. Pour simuler l'avance du robot, le panneau est avancé de 5 cm entre chaque prise d'image. Pour cette modélisation, 40 paires d'images stéréoscopiques sont prises, ce qui totalise une distance couverte de 2 m.

La boîte métallique et la boîte de plastique permettent de tester certaines hypothèses comme la contrainte de saturation pour le classement des observations et la corrélation de l'intensité entre deux surfaces lors de l'appariement.

5.2 Évaluation des performances

Il est important de mentionner que le calibrage est fait selon un montage peu précis et que les erreurs de mesure sont d'au plus 2 cm. Si on considère que la distance nominale de mesure est plus de 2,5 m, la précision sur la mesure est excellente. Il est difficile de comparer ces performances avec les capteurs existants car ceux-ci peuvent difficilement être utilisés dans les mêmes conditions. Néanmoins, même si certains capteurs existants offrent des précisions de l'ordre du millimètre, les performances actuelles sont amplement suffisantes pour l'application en cours.

Le tableau suivant indique la performance de certains critères du système de vision stéréoscopique.

Tableau IX

Mesure de performance du système de vision stéréoscopique

Critère du système de vision stéréoscopique	Mesure de performance
Erreur de classification des points laser	0
Erreur d'appariement	0
Précision sur les mesures	± 2 cm
Fréquence d'échantillonnage	Non disponible – traitement en temps différé[20]

[20] On évalue qu'environ 250 points par seconde peuvent être mesurés sur la scène avec le système informatique actuel. Cette estimation considère tout le processus du système de vision stéréoscopique à raison d'environ 5 paires d'images par seconde, de l'acquisition à la reconstruction des n points 3D. Cette estimation considère également des images typiques ayant une complexité moyenne, semblable à la scène illustrée ici.

La figure suivante montre deux angles de visualisation du nuage de points obtenu par cette expérience.

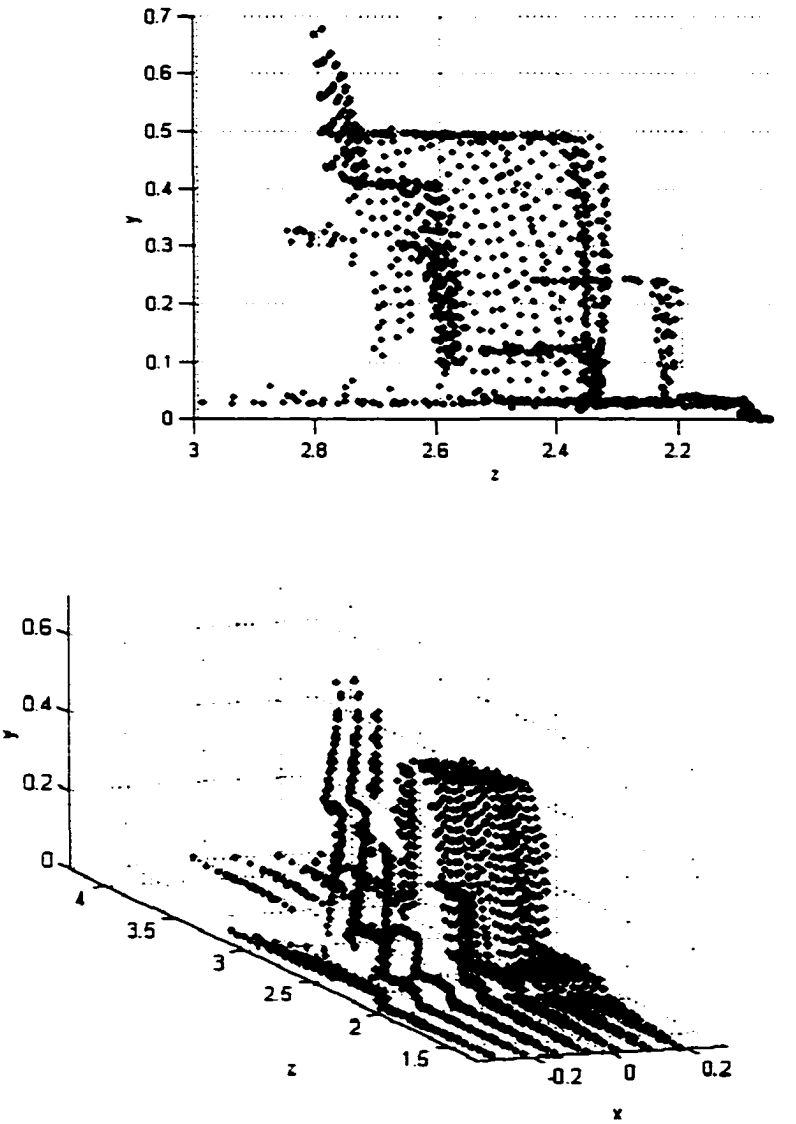


Figure 55 : Nuage de points de la scène modélisée

CONCLUSION ET RECOMMANDATIONS

L'objectif principal de cette première partie du mémoire est de développer un capteur permettant de faire un relevé topographique du sol. Le capteur proposé est un système de vision stéréoscopique binoculaire avec lumière structurée. La source lumineuse utilisée est un laser ayant un réseau de diffraction matricielle. Les avantages et les inconvénients de ce capteur permettent de comprendre les forces et aussi les limites de celui-ci. Les principaux avantages sont la quasi-indépendance du système à la complexité de la scène et aux conditions d'éclairage. Par contre, l'utilisation de la lumière laser cause deux désavantages importants. Premièrement, le laser est une source considérable de danger potentiel pour les gens qui côtoient le robot. Deuxièmement, les problèmes de réflexions secondaires des points laser peuvent causer de fausses représentations de l'environnement.

Tous les algorithmes implantés donnent une solution robuste au problème, même dans le cadre d'une application générique. Les scènes les plus complexes ayant beaucoup de bruits sur les images sont prises en considération par le classificateur. En effet, peu importe le contenu des images, l'algorithme du processus de vision trie les observations dans le but de retenir uniquement celles correspondant aux points laser. Le classificateur retenu est un réseau de neurones de type perceptron multicouche. Ce type de classificateur donne une erreur de classification de l'ordre de moins de $0{,}97 \pm 0{,}26$ %. Il ne faut pas perdre de vue que ces performances sont reliées à des images très complexes, ce qui est plutôt rare dans le cadre d'une application conventionnelle. Le temps de calcul requis pour classifier une observation avec le réseau de neurones est excellent, moins de 0,03 ms. À son tour, l'algorithme d'appariement donne des résultats plus que satisfaisants. Les tests faits sur des images d'une complexité moyenne révèlent une erreur d'appariement de 0 %.

À partir des contraintes physiques et des objectifs du projet, la surface couverte par le capteur est déterminée et est de forme trapézoïdale. La surface couverte va de 0,90 m à

3,50 m devant le robot. Elle couvre 2,50 m dans la partie la plus éloignée du robot et 1,19 m dans la partie la plus près. Avec une résolution spatiale de l'ordre de 10 cm, on peut couvrir une surface d'au moins 84 % pour l'avance du robot et d'au moins 59 % pour la rotation du robot. On mentionne aussi que ces pourcentages peuvent augmenter facilement à 100 % lorsque le laser est orienté convenablement.

Le prototype du capteur a permis de valider complètement tous les aspects du projet sur un montage statique. Avec ce montage, une précision meilleure que ± 2 cm sur les mesures est obtenue. Si on considère le calibrage rudimentaire fait, ces résultats sont satisfaisants si on tient compte que toutes les erreurs résultant de la chaîne de mesure sur le robot sont systématiquement plus grandes (l'erreur sur la mesure, l'erreur de position et d'orientation du capteur par rapport au robot et surtout l'erreur de position et d'orientation du robot par rapport à l'environnement).

Finalement, on peut dire que le capteur proposé remplit bien le mandat initial. Évidemment, plusieurs améliorations peuvent être proposées et doivent être prises en considération pour augmenter la fiabilité et les performances de l'appareil.

On propose plusieurs recommandations pour le capteur extéroceptif présenté.

a. La partie optique du système peut être améliorée ainsi.

    1. Le laser employé pour le prototype n'est pas assez puissant pour contrer les sources de lumière ambiante trop puissantes. Par exemple, le système n'est d'aucune utilité sur une scène extérieure en plein jour et en plein soleil. Le robot est tout simplement aveugle car il est impossible de discerner les points laser sur les images. Ainsi l'usage d'un laser plus puissant permet de résoudre ce problème. Par contre, il faut encore prendre en considération l'aspect sécurité d'un tel laser.

2. La matrice de points laser n'est pas uniformément répartie sur la scène. La géométrie créée par l'angle de projection du laser par rapport au sol en est la cause. Il est probable que pour certaines applications, on doit considérer une matrice de diffraction dont les angles entre chaque faisceau soient mieux adaptés. Bien que ce type de produit n'est pas standard, il est possible d'en tirer profit surtout si on désire augmenter le nombre de points au sol.

3. La puissance du laser et la sensibilité des caméras devraient être ajustées en temps réel en fonction des conditions d'éclairages. La section 4.3.2.2 présente les grandes lignes pour réaliser un tel système d'ajustement.

b. On peut améliorer le processus algorithmique par les points suivants.

1. Les calculs des attributs de saturation et de ressemblance avec une distribution normale bivariante sont basés sur l'évaluation de l'intensité du fond de l'image entourant chacune des observations. Or, aucune méthode précise d'évaluation de l'intensité de fond n'est utilisée et les approximations donnent des valeurs non optimales. Avec une évaluation adéquate de l'intensité du fond, ces descripteurs de formes seraient beaucoup plus performants.

2. La segmentation automatique basée sur les propriétés des histogrammes bimodales ne permet pas de faire une bonne segmentation sur les images n'ayant pas une distribution bimodale. Ainsi il est difficile de trouver un seuil performant pour tous les types d'images. C'est pourquoi il serait important de se pencher sur le sujet et de trouver un algorithme permettant de solutionner le problème dans toutes les conditions puisque le capteur est développé dans un contexte d'application générique. La méthode de Otsu (1979) est certainement l'approche qui devrait être implantée. Cette méthode, basée sur la maximisation de la variance intra-classe permettrait une meilleure segmentation de l'image et dans un plus grand nombre de cas.

3. L'algorithme d'appariement reste performant seulement pour les petits groupes d'appariements. Dès que le nombre d'appariements potentiels augmente, l'algorithme devient inefficace. Les techniques de programmation dynamique permettant de solutionner élégamment l'algorithme de Viterbi vaudraient la peine d'être implantées.

4. Une autre recommandation concernant la partie algorithmique est relié à la lacune principale du système : les fausses représentations dues aux réflexions secondaires du projecteur laser. Malheureusement, la réalisation d'un algorithme permettant de solutionner ce problème est particulièrement complexe. Selon les connaissances disponibles, aucune méthode d'analyse locale des données ne permet d'identifier ces situations. Seulement une analyse globale des données pourrait donner une solution viable.

5. Un aspect qui n'a pas été exploité dans la conception du capteur est la connaissance des trajectoires des faisceaux laser. Il serait très facile de calculer à l'aide du système de vision stéréoscopique les droites paramétriques dans l'espace correspondant aux trajectoires des N faisceaux du projecteur laser. Cette connaissance supplémentaire du système optique permettrait de simplifier considérablement trois aspects du problème. Premièrement, on pourrait ajouter au processus de classification une contrainte de position en plus des contraintes de formes déjà utilisées. Une observation se trouvant ailleurs que sur une des trajectoires correspondant aux faisceaux laser ne pourrait être considérée comme étant un point laser. Deuxièmement, le problème de fausses représentations du aux réflexions secondaires du projecteur laser (mentionné au point précédent) serait réduit. Comme pour le problème de classification, un point laser se trouvant ailleurs que sur une des trajectoires des faisceaux laser indiquerait que ce point laser est issu d'une réflexion et non d'un des faisceaux. Finalement, on pourrait ajouter une contrainte supplémentaire à l'étape d'appariement

permettant de réduire encore le nombre de faux appariements. Il serait possible d'évaluer à quel(s) faisceau(x) laser(s) un point laser est susceptible d'appartenir : ainsi le point laser homologue dans l'autre image doit obligatoirement appartenir à l'un de ces faisceaux laser. L'utilisation de cette méthode serait grandement souhaitée surtout qu'elle est relativement facile à implanter. En fait, il suffirait de calibrer le système laser (cette étape consiste à évaluer l'équation des droites paramétriques correspondant aux faisceaux laser) à l'aide de N images présentant tous les points laser sur une surface plane. Cette étape devrait être faite après avoir calibré les caméras : ainsi il est possible de connaître N points par droites. En utilisant une méthode aussi simple que la méthode des moindres carrés on évaluerait précisément chacune des droites paramétriques.

c. L'implantation matérielle du prototype est loin d'être une solutions optimisée et rapide. Par contre, dans le cadre d'une application robotique, il est pertinent que le système puisse échantillonner des points de l'environnement le plus rapidement possible. D'ailleurs, si on considère un robot capable de se déplacer rapidement, le système de vision actuel ne permet pas un relevé topographique assez dense du terrain pour une planification de trajectoire efficace.

d. La liste des recommandations ne peut se terminer sans qu'il soit suggéré de faire l'évaluation du système sur un mobile. En fait, cette étape est essentielle pour évaluer les performances globales de cette approche.

DEUXIÈME PARTIE

REPRÉSENTATION VIRTUELLE DE L'ENVIRONNEMENT

INTRODUCTION

Le rôle du module de représentation virtuelle de l'environnement (RVE) est de représenter l'espace réel à l'aide d'une structure de données informatique utilisable par le module de la planification de trajectoire. Pour le travail présenté dans ce mémoire, ce module, appelé aussi carte virtuelle, est l'interface reliant les modules de la perception de l'environnement et de la planification de trajectoire.

Les entrées du module RVE dépendent grandement des types d'applications destinés au robot et encore plus des méthodes utilisées pour rendre ce dernier autonome. Néanmoins, on y retrouve principalement les capteurs extéroceptifs, les connaissances à priori sur l'environnement et le système de positionnement.

Les capteurs extéroceptifs sont les capteurs permettant de percevoir l'environnement. Cette perception est essentielle pour augmenter les connaissances sur l'environnement lorsqu'une des trois situations suivantes se présentent :

a. un environnement partiellement ou totalement inconnu;

b. un environnement dynamique (où on considère le déplacement de certains acteurs de la scène dans le temps);

c. un système de positionnement absent ou peu précis. Dans ce cas, certaines mesures doivent être prises pour que le robot puisse se déplacer convenablement (passer par une porte étroite demandant une certaine précision de positionnement par exemple).

Les connaissances à priori permettent au robot de *connaître* l'environnement dans lequel il doit se mouvoir. Avec cette connaissance, il est possible de s'orienter dans la bonne direction dès les premiers instants. De plus, tel que présenté par Castellanos et Tardos (1999) et bien d'autres, il est aussi possible de connaître l'endroit où se trouve le robot si ce dernier possède un système de perception de l'environnement suffisamment précis pour permettre la mesure d'un espace fini et pour en faire la correspondance avec un

élément de la RVE. D'un autre côté, le système de positionnement permet de connaître la position relative ou absolue du robot par rapport à un référentiel. Cette connaissance est très utile car elle valide la position du robot pour le module de suivi de la trajectoire. Elle permet aussi l'insertion de nouvelles données spatiales dans la carte virtuelle. Tous ces outils servent à utiliser la RVE, non seulement comme interface entre la perception de l'environnement et la planification de trajectoire mais aussi comme une mémoire spatiale de l'environnement permettant une meilleure planification de trajectoire à tout moment.

À la sortie du module de RVE, on possède une ou plusieurs cartes virtuelles sur lesquelles on doit retrouver une représentation synthétisée de toutes les données à l'entrée. Cette synthèse permet une compréhension suffisante du monde réel pour donner les informations nécessaires au module de planification de trajectoire. Ainsi la forme que prennent les données à la sortie dépend grandement du type de planificateur de trajectoire. Néanmoins, peu importe le type d'algorithme utilisé par le module de planification de trajectoire, le rôle commun le plus important de toutes les RVE est de bien identifier les endroits où le robot peut circuler et les obstacles qu'il ne peut franchir.

Il existe un autre aspect commun pour tous les types de RVE, c'est la minimisation de la complexité de la structure de données. L'espace mémoire requis est parfois très grand : c'est pourquoi on cherche à utiliser une structure de données permettant une représentation optimale dans une taille minimale de mémoire tout en gardant un accès aux données dans un délai minimum.

L'objectif de cette deuxième partie n'est pas de couvrir tous les domaines associés aux RVE puisque ceux-ci sont très vastes et s'étendent sur des concepts allant au-delà de ceux abordés dans ce mémoire. Le but est d'examiner les principales RVE existantes et de présenter la solution réalisée.

Le chapitre 6 présente les types de RVE les plus couramment utilisés. On met l'accent sur ceux qui peuvent le mieux représenter les données en fonction des besoins du présent système. Ce chapitre commence par une brève revue de littérature et présente ensuite la définition du problème ainsi que la solution implantée. On y aborde aussi le problème d'insertion des données dans la carte virtuelle. Le chapitre 7 présente en détail le processus algorithmique permettant de solutionner le problème.

CHAPITRE 6

DÉFINITION DE LA PROBLÉMATIQUE

6.1 Revue de littérature

Puisque le robot marcheur est un robot terrestre, on présente dans ce mémoire les RVE qui considèrent seulement les environnements de type bidimensionnels (2D) ou bidimensionnels et demi (2D½). Les termes 2D et 2D½ indiquent des environnements décrits par des surfaces. Le terme 2D indique des surfaces planes et 2D½ des surfaces ayant différentes élévations pour lesquelles le chemin le plus court entre deux points n'est pas une droite. On ne considère donc pas des environnements 3D utiles pour les robots aériens ou aquatiques.

La figure suivante représente deux scènes typiques. La première correspond à un environnement structuré sur un terrain plat (2D) et le deuxième représente un terrain extérieur (2D½).

a) Scène d'un terrain 2D b) Scène d'un terrain 2D½

Figure 56 : Scènes typiques sur terrain 2D et 2D½

6.1.1 La représentation exacte

La représentation exacte est le type de RVE de plus haut niveau. Elle consiste à décrire précisément la nature des acteurs de la scène. Elle offre la description la plus concise et aussi la plus précise. Malgré ses avantages incomparables, elle reste difficile à implanter et à utiliser, autant par la nature des données à l'entrée que par le module de planification de la trajectoire à la sortie. Même si ce type de représentation performe bien pour les environnements structurés, elle reste très difficile à utiliser pour les environnements non structurés.

Tableau X

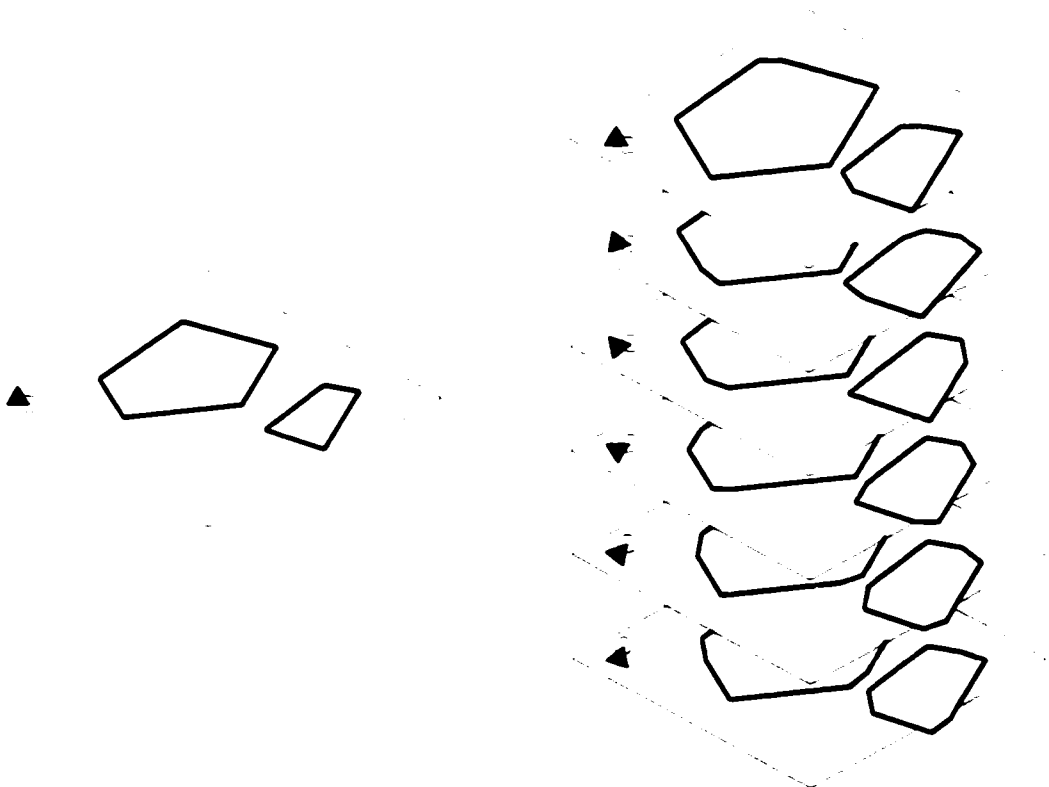
Exemples de représentation exacte

Scène 2D – Figure 56a *Environnement structuré*		**Scène 2D½ – Figure 56b** *Environnement non structuré*	
Scène	Pièce d'une maison de 100 × 100 (dm)	Scène	Scène extérieure de 100 × 100 (km)
Terrain	Plat	Terrain	Très difficile à décrire précisément
Obstacles	- Table circulaire de rayon 6 placée en (61, 59) - 4 chaises circulaires de rayon 3 placées en (72, 59) – (61, 48) – (61, 70) – (50, 59) - Un divan en coin de 19 × 19 placé en (89, 89) - Une causeuse 2 places de 59 × 19 placée en (70, 10) - Une bibliothèque en coin de forme triangulaire de 21 × 21 placée en (5, 94) - Une armoire rectangulaire de 18 × 43 placée en (9, 21)	Obstacles	Aucun

6.1.2 La représentation de l'espace libre

Cette forme de RVE considère les dimensions réelles du robot ainsi que les contraintes cinématiques de ce dernier. En fait, cette représentation consiste à appliquer une transformation de l'espace connu dans un espace à n dimensions où n correspond aux

différents degrés de liberté du robot. Par exemple, la plupart des robots peuvent être localisés dans l'espace par la coordonnée d'un point de référence *(x, y)* et par l'orientation du robot par rapport au référentiel de l'environnement. L'espace libre 3D résultant représente tous les endroits où le robot peut se situer sans collision, en considérant sa position et son orientation. La figure suivante montre en *a* un mobile de forme triangulaire ainsi que deux obstacles et, en *b*, l'espace libre vu selon 6 orientations du mobile (variant de 0 à 120°). Le cercle sur le mobile indique sa référence *(x, y)*. On remarque la déformation successive des obstacles devant indiquer la frontière pour le centre *(x, y)* du robot en fonction de chaque orientation.



a) Mobile et obstacles dans un plan

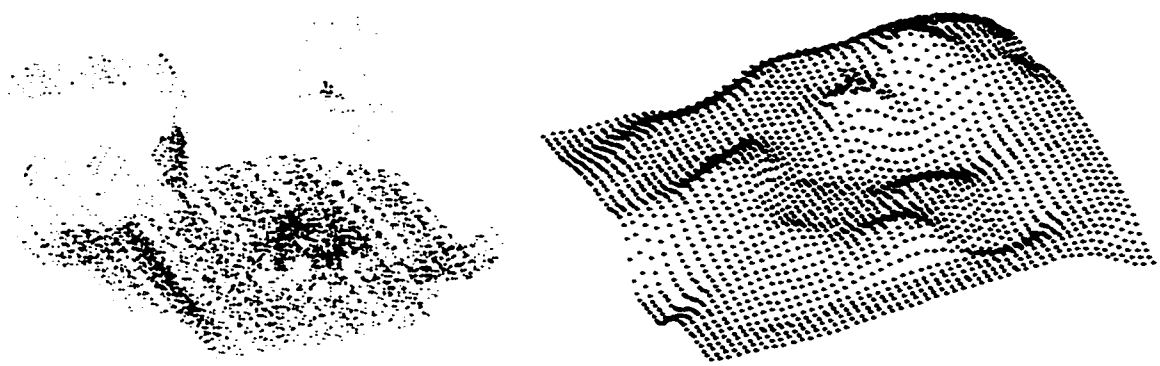
b) Espace libre

Figure 57 : Représentation par l'espace libre

Ce type de représentation est puissant et permet de trouver des solutions à des problèmes complexes à résoudre. Par contre, la définition de l'espace libre nécessite beaucoup de calculs.

6.1.3 La représentation sous forme de nuage de points

Cette forme de RVE est utile pour représenter les données brutes venant de certains capteurs extéroceptifs qui prennent des mesures ponctuelles de l'environnement. Pour s'en convaincre, il suffit de penser aux capteurs de type télémètre ou encore à celui présenté à la première partie. Malheureusement, la performance de ce type de représentation est partiellement limitée. Plusieurs raisons font en sorte qu'elle reste inadéquate. En premier lieu, les données brutes représentent tous les acteurs de la scène sans aucun discernement : le terrain et les obstacles font partie intrinsèque des mêmes données. Deuxièmement, la répartition des données est souvent mal distribuée et cette non-uniformité engendre fréquemment de nombreuses répétitions dans les données. Finalement, la liste de points n'assure pas un accès rapide aux données. Il faut régulièrement balayer l'ensemble des données pour accéder à une seule coordonnée spatiale.

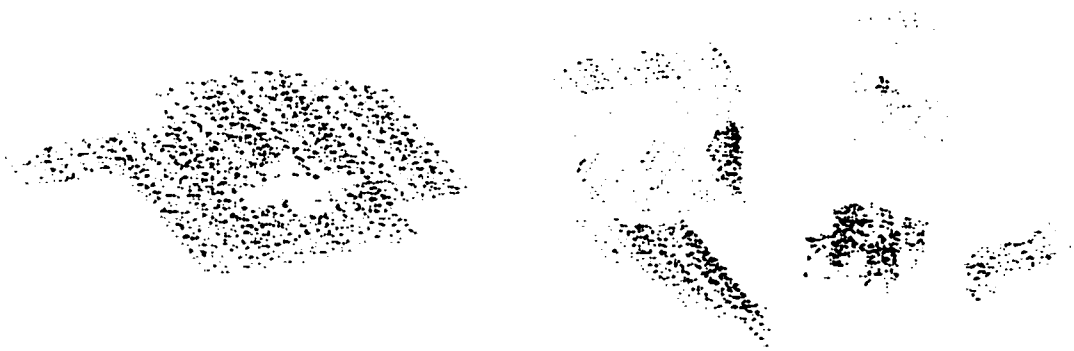


a) Nuage de points de la scène 2D (figure 56a)

b) Nuage de points de la scène 2D½ (figure 56b)

Figure 58 : Exemples de représentation sous forme de nuage de points

L'analyse des données nécessaires pour déterminer les points correspondant au sol et aux obstacles diffère grandement si la scène est de type 2D ou de type 2D½. En effet, dans un environnement structuré, le sol est constitué d'une suite de plans horizontaux ou légèrement inclinés. Lorsque la hauteur du sol est connue, il est facile d'identifier tous les points correspondant à ce dernier. Tous les autres points représentent les obstacles. Par contre, il est impossible d'utiliser la même hypothèse pour un environnement non structuré puisque tous ses éléments peuvent prendre toutes les formes possibles. Ainsi il devient plus difficile de déterminer où se trouvent les obstacles sur la scène. La figure suivante montre la séparation du nuage de points pour le sol (a) et pour les obstacles (b) de la figure 56a.



a) Nuage de points du sol b) Nuage de points des obstacles

Figure 59 : Séparation du nuage de points pour une scène 2D (terrain et obstacle)

6.1.4 Surface d'élévation et volume d'occupation

Ces méthodes consistent à diviser l'environnement en petites cellules de même taille. La première RVE est la surface d'élévation. Elle utilise une surface plane divisée en $m \times n$ cellules surfaciques. Chaque cellule est bornée dans l'espace et permet d'indiquer l'élévation maximum de la scène à cet endroit. Cette technique est en fait une projection de l'espace tridimensionnel sur le plan orthogonal x-y (z étant la hauteur). Cependant une somme importante d'informations est perdue. Ainsi certaines formes complexes de

l'environnement peuvent générer des RVE ne permettant pas à un véhicule de circuler alors qu'il existe réellement un espace suffisant. Ces situations se présentent principalement lorsque l'environnement possède des volumes ayant des concavités définies sur l'axe des z. La figure suivante illustre ce propos. On retrouve une table accompagnée de quatre chaises dans un corridor étroit. La table et les chaises ont toutes la même forme, un pied cylindrique surmonté d'un plateau circulaire de plus grande dimension. À gauche, on retrouve une représentation physique de la scène et au centre, on retrouve la surface d'élévation correspondante. La situation suivante illustre ce fait. Un robot de petite taille dont la hauteur est inférieure à la hauteur du plateau circulaire de la chaise doit faire le nettoyage. L'erreur causée par l'approximation de la RVE ne montre aucun chemin possible alors qu'il en existe vraiment un.

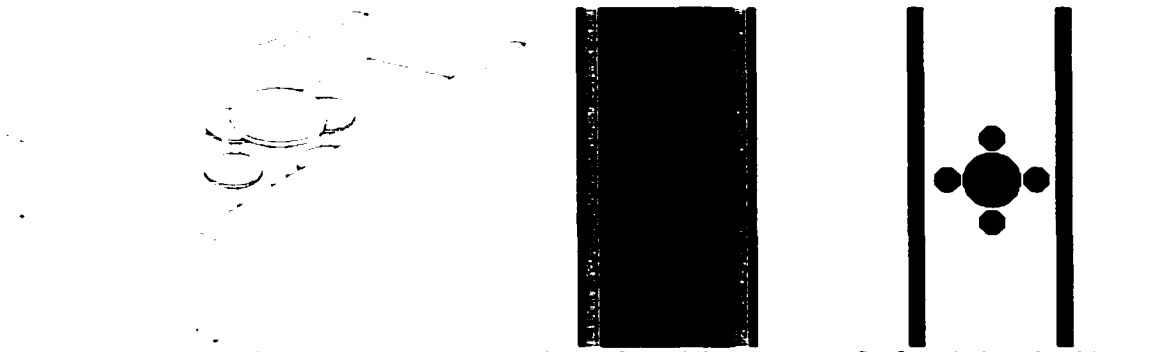


Figure 60 : Problématique reliée à la surface d'élévation

La deuxième RVE présentée permet de corriger cette limitation. Le volume d'occupation consiste à diviser un volume en $m \times n \times p$ cellules volumiques. Cette approche permet de représenter toutes les composantes de la scène de façon discrète. Contrairement à la surface d'élévation, le volume d'occupation nécessite seulement 2 bits par cellule puisque seulement 4 types d'informations sont nécessaires : cellule dont le contenu est inconnu, cellule vide, cellule partiellement pleine et cellule pleine.

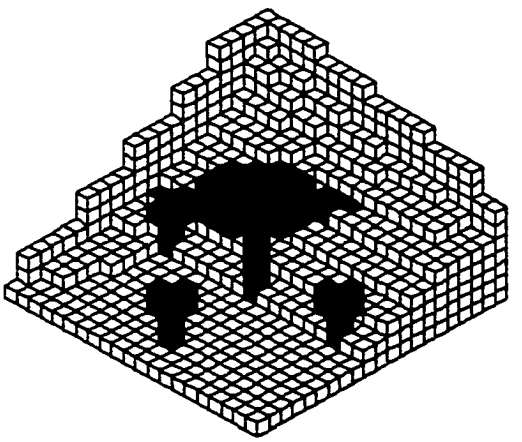


Figure 61 : Exemple d'un volume d'occupation (centre de la figure 56a)

Contrairement au nuage de points, la RVE de type surface d'élévation ou volume d'occupation nécessite toujours la même quantité d'espace mémoire. Cette technique n'est pas optimale si on considère l'espace mémoire requise. Par exemple, une surface d'élévation couvrant une salle de 10 × 10 m avec 1 cm de résolution spatiale et avec 16 bits de résolution par élément de surface requiert 1 000 000 de cellules. Avec 2 octets (16 bits) pour représenter chacune des cellules, c'est plus de 15,2 Mo qui sont nécessaires. Si on désire représenter un terrain de grande dimension avec un volume d'occupation, la quantité de mémoire requise devient énorme. Par exemple, plus de 37,2 Go sont nécessaires pour représenter un terrain de 1 × 1 × 0,1 km à une résolution de 0,1 m.

6.1.5 Autres types de décomposition et de représentation compacte

Les méthodes de décomposition cellulaire vues à la section précédente sont très utilisées et nécessitent des techniques de représentation compacte des données. Il existe dans la littérature plusieurs techniques permettant de compresser les données de ces RVE. On présente ici les plus importantes et les plus répandues. Le lecteur intéressé peut se référer à Pruski (1996).

Une technique souvent utilisée consiste à appliquer une division rectangulaire verticale (ou horizontale) de l'espace en fonction des obstacles se trouvant dans l'environnement. La taille de chaque rectangle est maximisée pour prendre la plus grande surface sans obstacle. Pour chaque élément de la division, on recommence le processus récursivement jusqu'à ce que la résolution désirée soit obtenue.

Dans le but d'optimiser l'efficacité de la structure de données, on utilise une division de type binaire. Cette technique, nommée décomposition binaire, est un algorithme récursif de décomposition permettant une grande économie de l'espace mémoire. On divise la carte représentant les obstacles en quatre cellules distinctes de même taille. Pour chacune de ces cellules, on identifie la classe d'appartenance : cellules vides, partiellement pleines ou pleines. Temporairement, on divise en quatre chacune des cellules partiellement pleines. Si aucune des sous-divisions n'est vide, on laisse la cellule parent non divisée. Par contre, si une des sous-cellules temporaires est vide, on applique formellement la division cellulaire du parent et on reprend récursivement l'algorithme pour chaque nouvelle cellule partiellement vide jusqu'à ce que le seuil de résolution désiré soit atteint. Cette division s'effectue par groupe de quatre lorsqu'elle est appliquée à une représentation surfacique plane mais par groupe de huit lorsqu'elle est utilisée pour une représentation volumique.

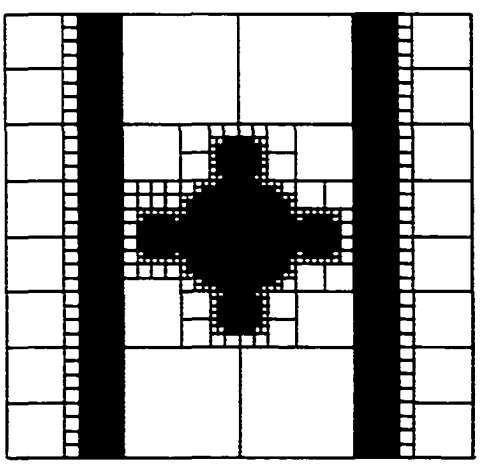


Figure 62 : Décomposition binaire de la figure 60a

Une méthode efficace permettant de représenter la RVE résultante de la décomposition binaire est l'utilisation d'un arbre 4-aire pour les représentations 2D et 8-aire pour les représentations 3D. Chaque niveau de l'arbre représente une étape de la division successive. La figure suivante est un exemple d'arbre quaternaire (*quadtree*). Il représente les quatre premiers niveaux de la figure 62. On remarque que les noeuds de l'arbre peuvent prendre les valeurs vides (blanches), partiellement pleines (grises) et pleines (noires). Les noeuds fils représentent, de gauche à droite, les cellules divisées selon l'ordre suivant : en haut à gauche (NO), en haut à droite (NE), en bas à gauche (SO) et finalement en bas à droite (SE) de la division.

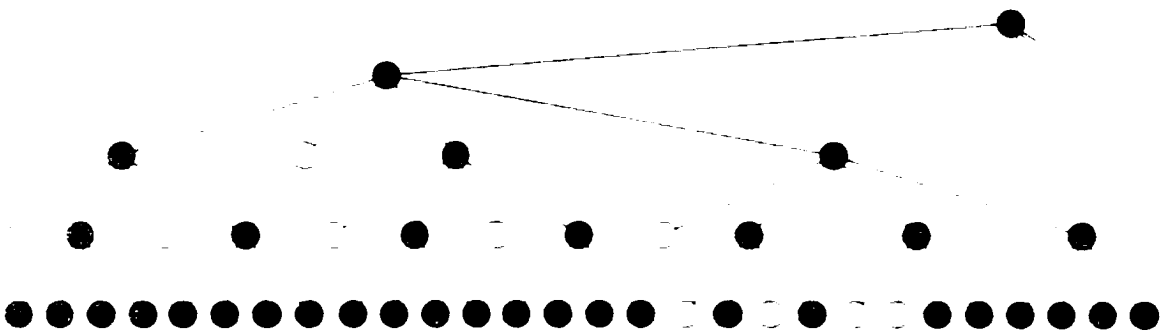


Figure 63 : Arbre de représentation quaternaire (*quadtree*)

Il existe un nombre incroyable de variantes autour du même thème et plusieurs façons de coder l'arbre binaire. Une technique intéressante consiste à coder chaque feuille de l'arbre en fonction de sa position dans la grille binaire de profondeur n. On attribue un poids différent pour chaque type de noeuds fils, c'est-à-dire :

$$P = \begin{cases} 0 \text{ si NO} \\ 1 \text{ si NE} \\ 2 \text{ si SO} \\ 3 \text{ si SE} \end{cases}$$

Le code de chaque feuille se calcule maintenant ainsi :

$$C = \sum_{i=0}^{n} P_i 4^i \tag{6.1}$$

où : C est le code d'une feuille

i est le niveau dans l'arbre (0 étant le niveau des feuilles et n le niveau de la racine)

P_i est le poids d'un noeud

Par exemple, le code de la dernière feuille en bas à droite de la figure 63 se calcule ainsi : $C = 4^3 + 0 \cdot 4^2 + 4 \cdot 4^1 + 4 \cdot 4^0 = 64 + 0 + 16 + 4 = 84$

6.1.6 Description polygonale

Cette technique de RVE est une méthode dérivée de la technique de l'espace libre. La description polygonale représente les obstacles définis par leur contour. Plusieurs algorithmes de planification de trajectoire sont basés sur ce type de description. Il suffit de décrire les obstacles en reliant les sommets qui décrivent le plus précisément possible le contour. Cette technique est très efficace pour approximer un obstacle au contour linéaire mais moins efficace pour le contour curviligne.

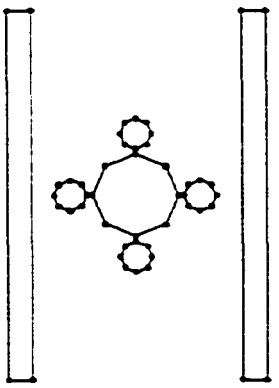


Figure 64 : Description polygonale des obstacles de la figure 60a

6.1.7 Description à l'aide de primitives géométriques

Le manque d'exactitude causé par l'approximation de certaines méthodes peut causer, dans les cas limites, des collisions avec les obstacles ou des recherches vaines pour trouver un chemin existant. La méthode de RVE à l'aide de primitives géométriques donne une solution élégante pour les environnements structurés pouvant facilement se décrire par des formes simples. Les primitives géométriques utilisées sont de même dimension que la RVE : on utilise des cercles et des carrés pour décrire une scène 2D et des sphères et des cubes pour décrire une scène 3D. Il est important de ne pas perdre de vue que cette méthode reste également approximative pour les environnements non structurés.

Tableau XI

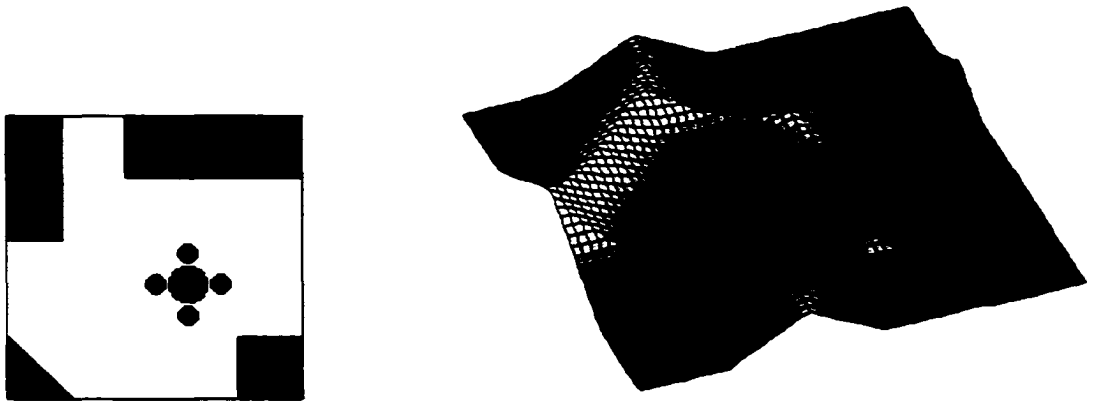
Exemples de représentation à l'aide de primitives géométriques

Scène 2D – Figure 56a *Environnement structuré*	**Scène 2D½ – Figure 56b** *Environnement non structuré*
- Cercle de rayon 6 centré à (61, 59) - 4 cercles de rayon 3 centrés à (72, 59), (61, 48), (61, 70) et (50, 59) - Un rectangle borné par (0, 0)-(18, 43) - Un rectangle borné par (80, 80)-(100, 100) - Un rectangle borné par (20, 0)-(100, 20) - Un triangle borné par (0, 78)-(21, 79)-(0, 100)	Impossible à décrire précisément.

6.1.8 Représentation par la proximité des obstacles

Cette RVE est introduite ici pour un type de planification de trajectoire spécifique : l'approche par champs de potentiels artificiels. Cette représentation est une transformation de l'espace des surfaces d'élévation ou des volumes d'occupation en un

espace indiquant la distance des obstacles les plus près. À chaque position de la surface ou du volume, on calcule simplement la distance de l'obstacle le plus près. On utilise, la distance euclidienne ou le carré de la distance pour accélérer les calculs. La figure suivante montre une image binaire des obstacles de la figure 56a et sa représentation par la proximité des obstacles.



a) Image binaire des obstacles b) Représentation par la proximité des obstacles

Figure 65 : Représentation par la proximité des obstacles de la scène 56a

6.2 Approche proposée

Avant de définir la problématique, il est important de répondre aux deux questions suivantes : quelles sont les motivations poussant à connaître l'environnement? et quelles parties de l'environnement sont intéressantes? Il existe deux réponses à la première question. La première motivation, justifiant amplement les besoins d'une RVE, est la nécessité de connaître l'environnement pour pouvoir y naviguer de façon intelligente. Le désir de performance des robots modernes nécessite des méthodes sophistiquées de planification de trajectoire car il ne suffit pas de naviguer aveuglément ou à tâtons pour trouver son chemin. La deuxième motivation est inhérente au robot marcheur. Étant donné que ce type de robot offre des capacités supplémentaires de déplacement en terrain accidenté, il importe de pouvoir identifier les endroits où le robot peut déposer ses pieds pour augmenter l'efficacité de sa démarche. La réponse de la première

question amène la réponse de la deuxième question. Il y a deux parties différentes du terrain à connaître : les obstacles et la forme du relief au sol. Désirer connaître exclusivement ces deux parties de l'environnement est la conséquence directe des hypothèses formulées au début du mémoire.

L'approche proposée est basée d'abord et avant tout sur la définition d'un obstacle : tout ce qui s'oppose au passage et/ou qui gêne le mouvement. Si on compare le robot marcheur aux autres robots terrestres, sur roues ou sur chenilles, on remarque sa capacité supplémentaire de franchir des obstacles insurmontables pour les autres. Trois types d'obstacles sont considérés pour le robot marcheur : les discontinuités verticales d'une certaine hauteur, les pentes trop abruptes[21] et les discontinuités horizontales[22] trop larges (correspondant aux crevasses).

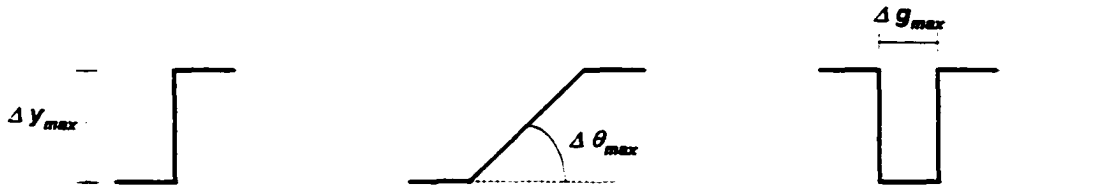


Figure 66 : Type d'obstacles pour un robot marcheur

On remarque que ces trois types d'obstacles en sont seulement s'ils dépassent une certaine grandeur. Par exemple, un marcheur peut utiliser un escalier car cet obstacle correspond à une discontinuité verticale où $\Delta y < \Delta y_{max}$. Tous les autres types de robots terrestres n'offrent pas autant de flexibilité.

---

[21] Les pentes sont considérées seulement sur une certaine surface et non en un seul point car les pentes ponctuelles doivent être considérées comme étant des discontinuités verticales.

[22] Les obstacles de type discontinuités horizontales n'ont pas été implantés pour l'élaboration du prototype.

Puisque la RVE est en quelque sorte l'interface entre le module de perception de l'environnement et le module de planification de trajectoire, on a, à l'entrée, un nuage de points et, à la sortie, deux cartes permettant de représenter les deux types d'informations désirées. La première carte contient simplement le relevé topographique du sol sous forme d'une carte d'élévation. La deuxième carte présente les obstacles sous forme d'une carte d'obstacles (qui est une carte d'élévation binaire). Le schéma suivant montre le processus algorithmique développé.

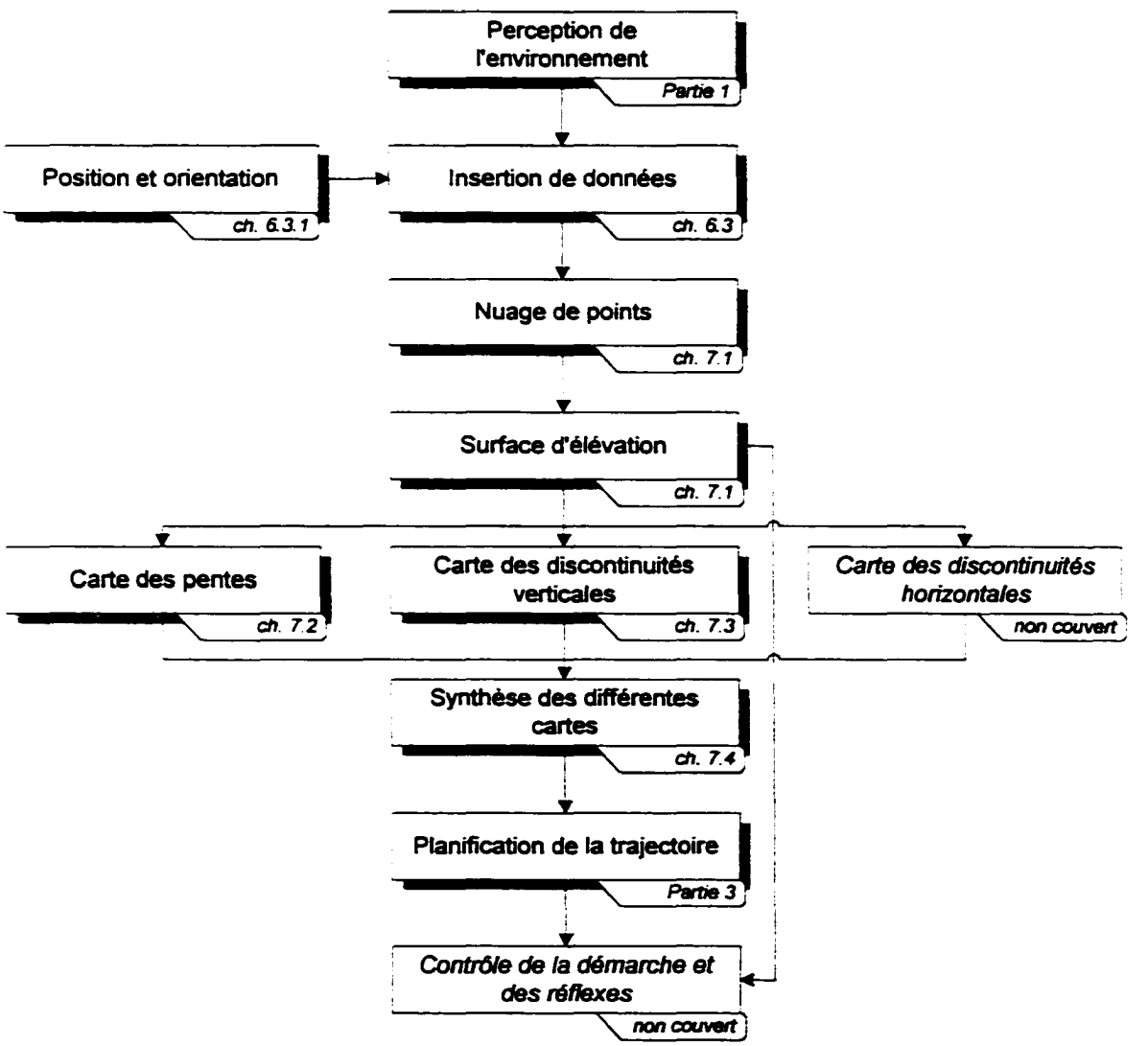


Figure 67 : Processus algorithmique de la RVE

6.3 Insertion des données

Le problème d'insertion de données dans la carte virtuelle se divise en trois parties : connaître la position et l'orientation du robot par rapport au référentiel de l'environnement, appliquer une transformation aux coordonnées spatiales des données pour qu'elles soient en fonction du référentiel de l'environnement et prendre en considération les erreurs de mesure.

6.3.1 Position et orientation du robot

La position et l'orientation du robot par rapport au référentiel de l'environnement doivent être connues. Plusieurs techniques permettent de mesurer la position et l'orientation d'un mobile dans l'espace. Chacune offre des avantages et des inconvénients. Ce sujet n'est pas couvert dans ce mémoire mais on peut citer à titre d'exemple quelques techniques qui existent. Le GPS permet, par le biais de balises géostationnaires, de connaître la position absolue d'un mobile. Cette technique très puissante possède néanmoins l'inconvénient de fonctionner uniquement dans les endroits extérieurs et découverts. Par contre, sous certaines conditions, elle peut offrir une très grande précision. Le « *dead reckoning* » est une technique permettant d'évaluer la position et l'orientation relatives du robot en comptant le déplacement que ce dernier a fait depuis son origine. On calcule le nombre de tours de roue pour un robot rouleur et le déplacement des pattes au sol pour un marcheur. Même si cette technique reste simple à implanter, elle offre de mauvaises performances à cause des problèmes de glissement et d'intégration. Une autre approche intéressante est l'usage d'une centrale inertielle. Cette technique est basée sur l'intégration de l'accélération du corps du mobile pour connaître la position, la vitesse et l'accélération linaire et angulaire du robot. Cette technique, très prometteuse, offre quand même une grande dérive dans le temps due au problème d'intégration. Plusieurs autres techniques existent et le lecteur intéressé peut se référer à Everett (1995) et à Pruski (1996).

Pour le développement des étapes subséquentes, on suppose connues la position et l'orientation du robot.

6.3.2 Transformation des coordonnées spatiales

Les capteurs extéroceptifs donnent les positions selon le référentiel du capteur. Or ce référentiel est référencé par rapport au référentiel du robot qui est lui-même référencé par rapport au référentiel de l'environnement. Puisque la RVE est référencée selon le référentiel de l'environnement, il est essentiel de faire la transformation des coordonnées dans le bon référentiel.

On utilise la technique des coordonnées homogènes pour faire cette transformation. Soit deux référentiels *R1* et *R2*, on connaît la grandeur de la translation (t_x, t_y, t_z) et de la rotation $(\theta_x, \theta_y, \theta_z)$ qui existent entre *R1* et *R2*.

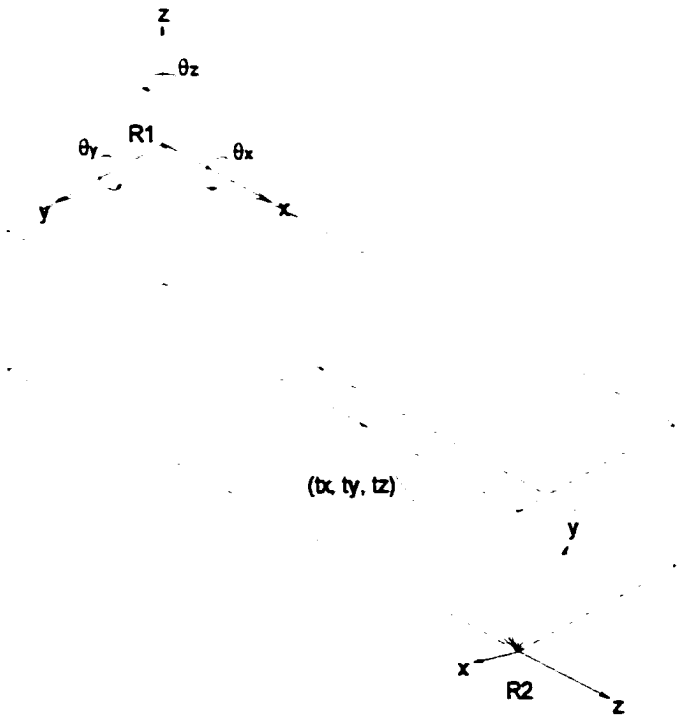


Figure 68 : Transformation des coordonnées spatiales entre deux référentiels

La matrice de transformation est définie comme suit :

$$T = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_x \\ r_{12} & r_{23} & r_{32} & t_y \\ r_{13} & r_{23} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

où : T est la matrice de transformation homogène

t est le vecteur de translation

r est la matrice de rotation 3 × 3

$$r = \begin{bmatrix} \cos\theta_z \cos\theta_y & \cos\theta_z \sin\theta_y \sin\theta_x - \sin\theta_z \cos\theta_x & \cos\theta_z \sin\theta_y \cos\theta_x + \sin\theta_z \sin\theta_x \\ \sin\theta_z \cos\theta_y & \sin\theta_z \sin\theta_y \sin\theta_x + \cos\theta_z \cos\theta_x & \sin\theta_z \sin\theta_y \cos\theta_x - \cos\theta_z \sin\theta_x \\ -\sin\theta_y & \cos\theta_y \sin\theta_x & \cos\theta_y \cos\theta_x \end{bmatrix}$$

On calcule la matrice de transformation globale correspondant à la multiplication des matrices de transformation du capteur au robot et ensuite du robot à l'environnement. On calcule les coordonnées du point selon le nouveau référentiel ainsi :

$$\begin{bmatrix} \frac{x_e}{s} \\ \frac{y_e}{s} \\ \frac{z_e}{s} \\ s \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \cdot T_{cr} \cdot T_{re} \tag{6.3}$$

où : (x_e, y_e, z_e) est la coordonnée relative au référentiel de l'environnement

(x_c, y_c, z_c) est la coordonnée relative au référentiel du capteur

T_{cr} est la matrice de transformation du référentiel du capteur au référentiel du robot

T_{re} est la matrice de transformation du référentiel du robot au référentiel de l'environnement

6.3.3 Erreurs sur les mesures

Sur chaque mesure prise par les capteurs extéroceptifs, il existe plusieurs sources d'erreur. De plus, la synthèse des informations occasionne un accroissement important des erreurs de mesure. Il suffit de penser à la transformation des coordonnées d'un repère à l'autre pour s'en convaincre. Les deux transformations subséquentes ajoutent les incertitudes de l'évaluation des positions et des orientations.

Il existe dans la littérature plusieurs techniques de filtrage (comme le filtre de Kalman) et plusieurs techniques de RVE et de planification de trajectoire permettant de travailler avec des données incertaines. Puisqu'on n'aborde pas ce problème dans ce mémoire, le lecteur intéressé peut se référer à Ayache (1989), Castellanos et Tardos (1999) ainsi qu'à Dudek et Jenkin (2000) pour n'en nommer que quelques uns.

CHAPITRE 7

CONCEPTION ET DÉVELOPPEMENT ALGORITHMIQUE DE LA SOLUTION

7.1 Nuage de points et surface d'élévation

On pose P, le vecteur de points correspondant au nuage de points obtenus suite à l'insertion des mesures ponctuelles du capteur extéroceptif dans la RVE et SE la surface d'élévation de dimension $m \times n$ dont les dimensions définissent la résolution de l'espace discrétisé.

La tâche est simple : on initialise SE à une valeur inférieure au minimum des points inclus dans le vecteur P et on parcourt tous les éléments de P pour en faire l'insertion dans SE. Cette insertion se fait à condition que $P_{i_z} > SE_{P_{i_x},P_{i_y}}$.

La valeur d'initialisation correspond aux zones de l'environnement qui sont définies inconnues. Lors du remplissage du nuage de points, on insère les données et les cases correspondantes deviennent connues.

La figure suivante montrent la surface d'élévation selon différentes résolutions. On remarque que les zones inconnues sont illustrées par des barres dont l'amplitude est négative. Plus la résolution augmente, plus le nombre de cellules inconnues augmente. Par contre, l'analyse multirésolution reste possible et peut permettre de nombreux avantages.

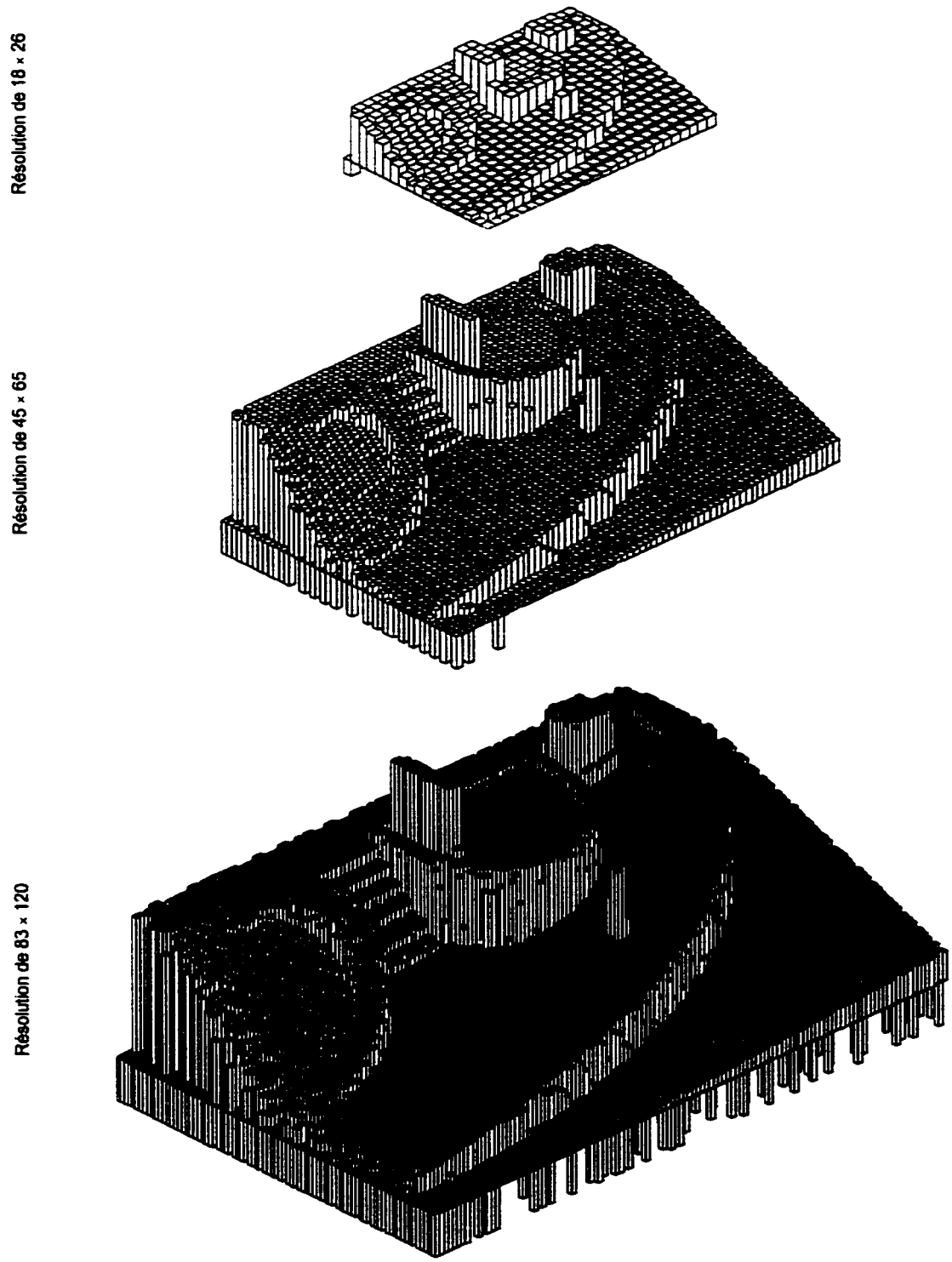


Figure 69 : Exemple de surfaces d'élévation (voir figure 135)

7.2 Carte des pentes

Déterminer où se trouvent les pentes trop abruptes de la scène limitant le déplacement du robot est une tâche délicate puisqu'il faut prendre en considération la direction de déplacement du robot. En effet, le robot peut se mouvoir sans problème sur des pentes raides si sa trajectoire n'est pas directement dans le sens le plus accentué.

On propose deux RVE : la première décrit les pentes par tranche successive (où chaque tranche indique une orientation précise) et la deuxième est la représentation de la pente maximale en tout point.

La pente est évaluée par le calcul de la dérivée première de l'image s'effectuant par la technique du gradient.

$$\nabla I = \frac{\partial I}{\partial x}\mathbf{i} + \frac{\partial I}{\partial y}\mathbf{j} \tag{7.1}$$

où : ∇I est le gradient de l'image

i et j indiquent les axes du vecteur directeur

On utilise l'opérateur de convolution appliquée sur l'image comme technique d'approximation rapide pour calculer le gradient.

$$\nabla I \approx I \oplus k \tag{7.2}$$

où : $\oplus$ est l'opérateur de convolution défini par :

$$I'_{x,y} = \sum_{j=-m}^{n}\sum_{i=-m}^{m} I_{x+i,y+j} \cdot k_{c+i,c+j}$$

k est le noyau de convolution

Voici les noyaux de convolutions les plus fréquemment utilisés :

$$k_v = \frac{1}{3}\begin{array}{|ccc|}\hline -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \\ \hline\end{array} \qquad k_h = \frac{1}{3}\begin{array}{|ccc|}\hline -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ \hline\end{array}$$

Figure 70 : Filtre de Prewitt

$$k_v = \frac{1}{4}\begin{array}{|ccc|}\hline -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \\ \hline\end{array} \qquad k_h = \frac{1}{4}\begin{array}{|ccc|}\hline -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \\ \hline\end{array}$$

Figure 71 : Filtre de Sobel

$$k_1 = \frac{1}{15}\begin{array}{|ccc|}\hline 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \\ \hline\end{array} \quad k_2 = \frac{1}{15}\begin{array}{|ccc|}\hline 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \\ \hline\end{array} \quad k_3 = \frac{1}{15}\begin{array}{|ccc|}\hline 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \\ \hline\end{array} \quad k_4 = \frac{1}{15}\begin{array}{|ccc|}\hline -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \\ \hline\end{array}$$

$$k_5 = \frac{1}{15}\begin{array}{|ccc|}\hline -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \\ \hline\end{array} \quad k_6 = \frac{1}{15}\begin{array}{|ccc|}\hline -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \\ \hline\end{array} \quad k_7 = \frac{1}{15}\begin{array}{|ccc|}\hline -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \\ \hline\end{array} \quad k_8 = \frac{1}{15}\begin{array}{|ccc|}\hline -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \\ \hline\end{array}$$

Figure 72 : Filtres directionnels de Kirsch

$$k_1 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \\ \hline\end{array} \quad k_2 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline 0 & -1 & -\sqrt{2} \\ 1 & 0 & -1 \\ \sqrt{2} & 1 & 0 \\ \hline\end{array} \quad k_3 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \\ \hline\end{array} \quad k_4 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline \sqrt{2} & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -\sqrt{2} \\ \hline\end{array}$$

$$k_5 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \\ \hline\end{array} \quad k_6 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline 0 & 1 & \sqrt{2} \\ -1 & 0 & 1 \\ -\sqrt{2} & -1 & 0 \\ \hline\end{array} \quad k_7 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \\ \hline\end{array} \quad k_8 = \frac{1}{2+\sqrt{2}}\begin{array}{|ccc|}\hline -\sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & \sqrt{2} \\ \hline\end{array}$$

Figure 73 : Filtres directionnels de Frei & Chen

On calcule la direction du vecteur orientation ainsi :

$$\varphi_{x,y} = \arctan\left(\frac{\nabla k_1 I_{x,y}}{\nabla k_2 I_{x,y}}\right) \tag{7.3}$$

où : $\varphi_{x,y}$ est l'orientation de la pente au point (x, y)

$\nabla_{k_n} I_{x,y}$ est le gradient de l'image au point (x, y) calculé selon le noyau k_n

k_2 est un noyau défini perpendiculairement au noyau k_1

L'amplitude de la pente se calcule ainsi :

$$\phi_{x,y} = \sqrt{\sum_{i=1}^{n} \left(\nabla_{k_i} I_{x,y}\right)^2} \tag{7.4}$$

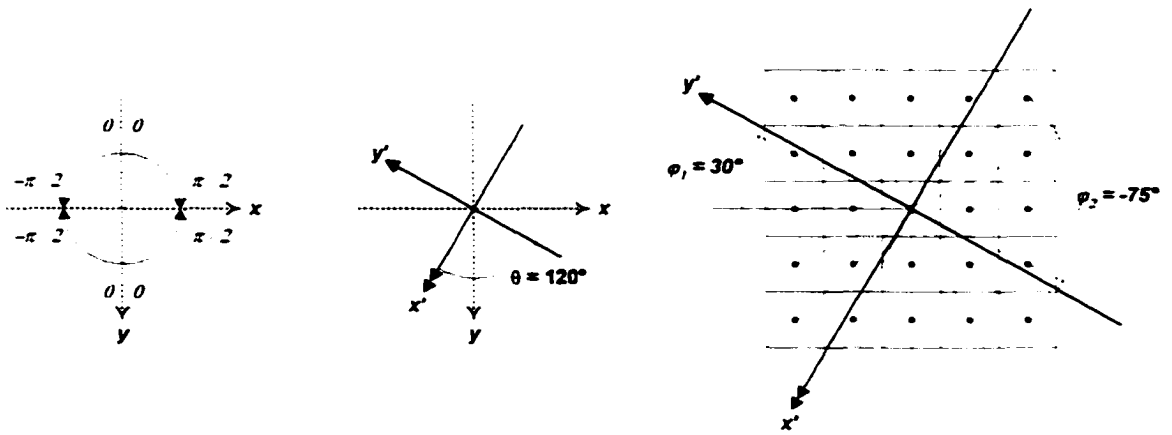
où : $\phi_{x,y}$ est l'amplitude de la pente au point (x, y)

i est le $i^{ième}$ filtre (jusqu'au $n^{ième}$) directionnel utilisé sur l'image

Malgré les performances intéressantes de ces filtres, on propose un nouveau filtre permettant de considérer les pixels du voisinage en fonction d'un angle d'intérêt spécifique. On fait référence à ce filtre par la méthode des directions. L'idée est simple : elle correspond à un filtre de premier ordre parfaitement orienté selon l'angle d'intérêt θ. Tous les pixels sont pondérés proportionnellement à l'angle de déphasage que leur centroïde fait par rapport à θ. Pour réaliser ce filtre, on définit un nouveau système d'axes. Le vecteur correspondant à l'angle d'intérêt indique l'amplitude maximum. Le vecteur opposé à l'angle indique l'amplitude maximum négative. Les directions perpendiculaires ont une valeur égale à 0 (la figure 74a illustre ce nouveau référentiel). Pour chaque orientation désirée, on superpose ce système d'axes sur le système d'axes conventionnel d'une image (l'axe des ordonnées étant orienté vers le bas) et on applique une rotation de ce système d'axes de la grandeur de θ (voir la figure 74b). Finalement, on calcule l'angle que forment les centroïdes des pixels en fonction

du nouveau système d'axes (voir la figure 74c). Pour normaliser l'amplitude du résultat selon la plage originale, on calcule le facteur de pondération λ par :

$$\lambda = \frac{1}{\pi \cdot \sum_{i=1}^{\frac{n-1}{2}} i} \tag{7.5}$$



a) Définition du nouveau système d'axes

b) Rotation du nouveau système d'axes

c) Évaluation des orientations selon le nouveau système d'axes

Figure 74 : Définition du noyau de convolution par la méthode des directions

Cette méthode a l'avantage de calculer rapidement l'amplitude des pentes pour une orientation spécifique. En effet, à l'aide d'un noyau de convolution défini par la méthode des directions, on a immédiatement l'amplitude de la pente selon le vecteur directionnel défini par θ. Ainsi l'orientation et l'amplitude se calculent rapidement et simplement ainsi :

$$\varphi_{x,y} = \theta \tag{7.6}$$

où : θ est l'angle du vecteur directionnel analysé (soit la direction du robot)

$$\phi_{x,y} = \nabla_{k_d} I_{x,y} \tag{7.7}$$

où : k_d est le noyau de convolution par la méthode des directions

La figure suivante illustre quelques exemples de filtres évalués selon la méthode des directions.

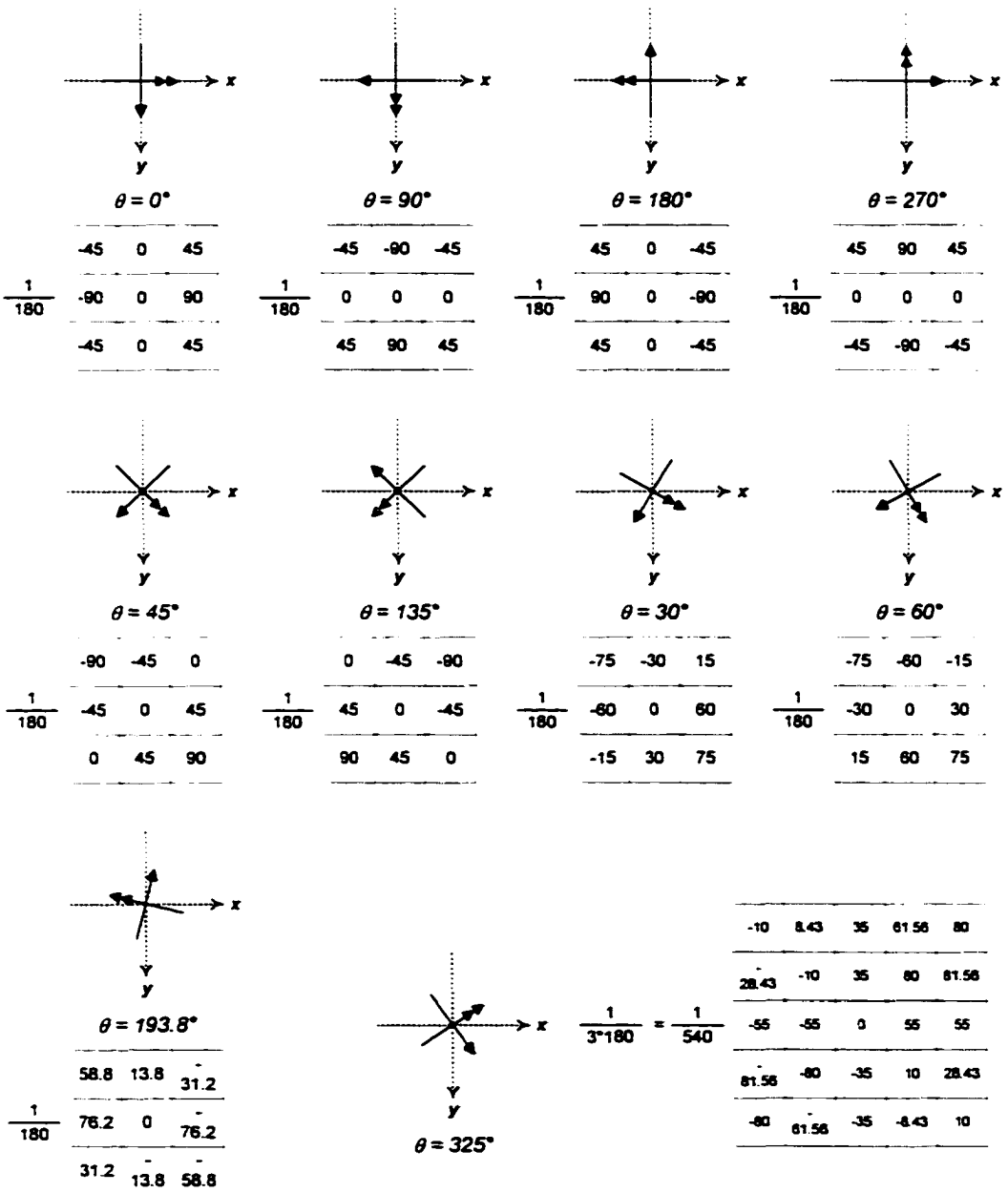
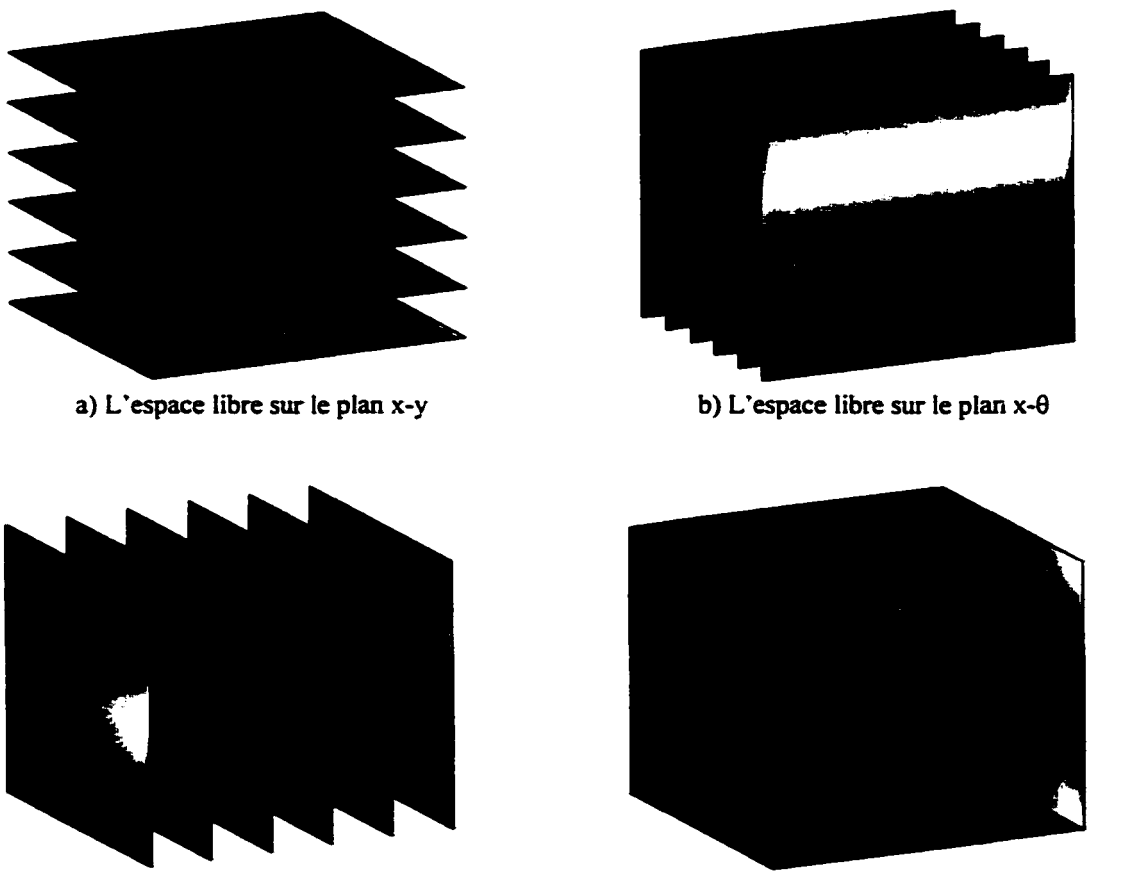


Figure 75 : Exemples de noyau de convolution par la méthode des directions

7.2.1 Représentation de l'espace libre selon toutes les directions

Cette représentation se fait par l'utilisation du filtre défini par la méthode des directions et par le calcul d'une carte différente pour chaque θ variant de 0 à 2π par un intervalle défini par δ. La figure suivante montre différents plans de vue du volume créé par cette représentation à partir de la figure 69 avec $\delta = \frac{\pi}{10}$.



a) L'espace libre sur le plan x-y

b) L'espace libre sur le plan x-θ

c) L'espace libre sur le plan y-θ

d) Vue de l'espace libre selon plusieurs coupes

Figure 76 : Représentation par couches successives

On segmente ensuite la carte en deux régions selon le seuil de tolérance θ_{max}. La fonction de seuillage est définie par :

$$CP_{x,y} = \begin{cases} 0 \text{ si } \nabla I_{x,y} > \theta_{max} \\ 0 \text{ si } \nabla I_{x,y} \leq \theta_{max} \end{cases} \quad (7.8)$$

La segmentation donne pour les figures 76a et 76d :

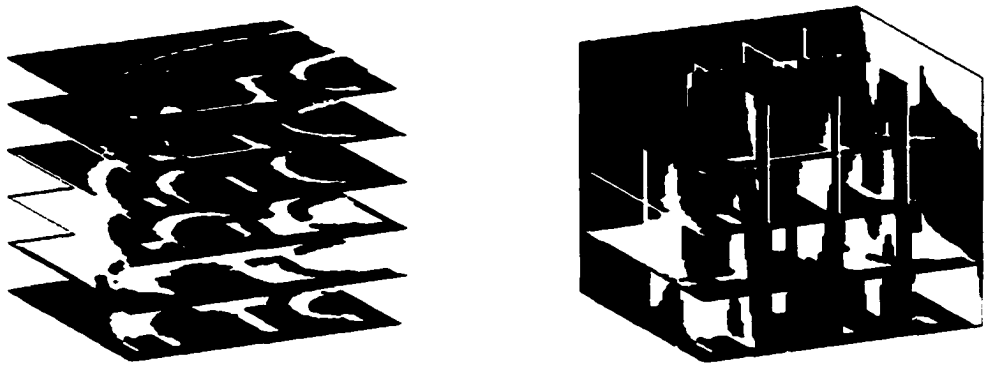


Figure 77 : Segmentation de la représentation par l'espace libre

En se référant à la définition d'un obstacle (voir figure 66), on remarque que les RVE ci-dessus indiquent les pentes et les discontinuités verticales. Par contre, on ne peut déterminer avec cette représentation si la pente couvre ou non une grande surface. Ainsi il est impossible de dire si une pente de 90° correspond à un mur que le robot ne peut franchir ou plutôt à un escalier fait de discontinuités verticales représentant à chaque marche une pente infinie. La prochaine section s'attarde à identifier ce type d'obstacle. Il faut cependant déterminer les endroits de grande surface correspondant à une pente régulière. Plusieurs techniques peuvent être employées, notamment des techniques d'analyse multirésolution mais l'approche présentée ici utilise plutôt un filtre morphologique de type ouverture sur l'image binaire. La première étape est une ouverture avec un élément structurant de petite taille permettant d'éliminer les régions très étroites comme les arêtes. La deuxième étape est l'ouverture avec un élément structurant de grande taille permettant d'identifier les grandes surfaces inaccessibles au robot. Les deux éléments structurants ont la forme d'un cercle. La taille du deuxième élément structurant est fonction de la résolution de la carte et de la taille relative du

robot. De plus, on ajoute un filtre médian pendant la première étape (entre la dilatation et l'érosion) pour éliminer les résidus ponctuels.

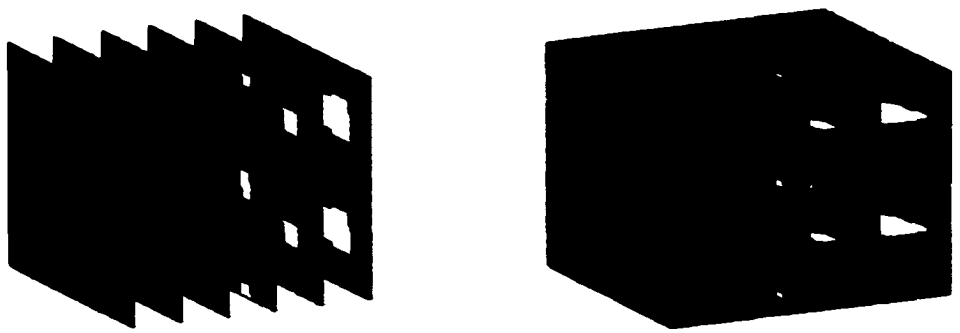


Figure 78 : Résultat de la première ouverture

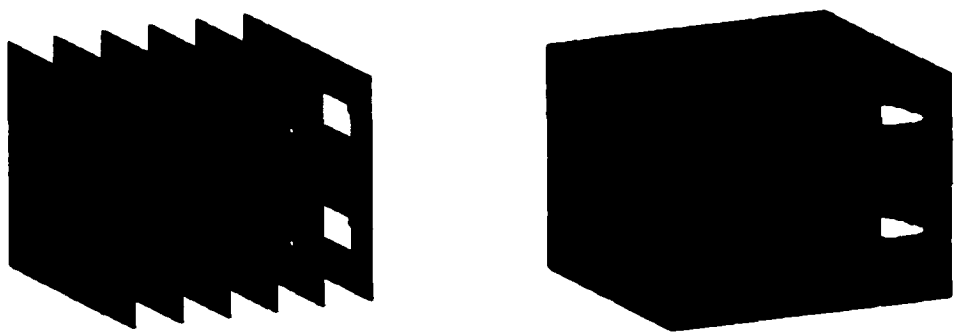


Figure 79 : Résultat de la deuxième ouverture

Voilà une bonne approximation de l'espace libre en fonction de l'inclinaison du terrain. On note que le noir indique l'espace libre et le blanc les inclinaisons trop prononcées sur les images binaires. La pente trop abrupte correspond à une région et à deux angles spécifiques, soit 90° et 270°. L'angle de 90° indique la trajectoire du robot lorsqu'il descend la butte et l'angle de 270° représente la trajectoire lorsqu'il grimpe la pente. On remarque que la représentation est symétrique sur les intervalles [0, π] et [π, 2π].

7.2.2 Représentation par les pentes maximums

Cette représentation est faite par le calcul de l'amplitude de la pente à l'aide du filtre de Kirsch. Cette méthode de représentation est beaucoup moins exhaustive que la

précédente et réduit considérablement l'espace libre. Cette réduction peut limiter grandement les mouvements du robot. Elle offre en revanche une plus grande simplicité de même qu'un modèle plus compact et plus facile à utiliser pour le module de planification de trajectoire.

Suite à l'application du filtre de Kirsch, on utilise la même technique que celle utilisée pour la représentation par l'espace libre : la segmentation et les deux ouvertures.

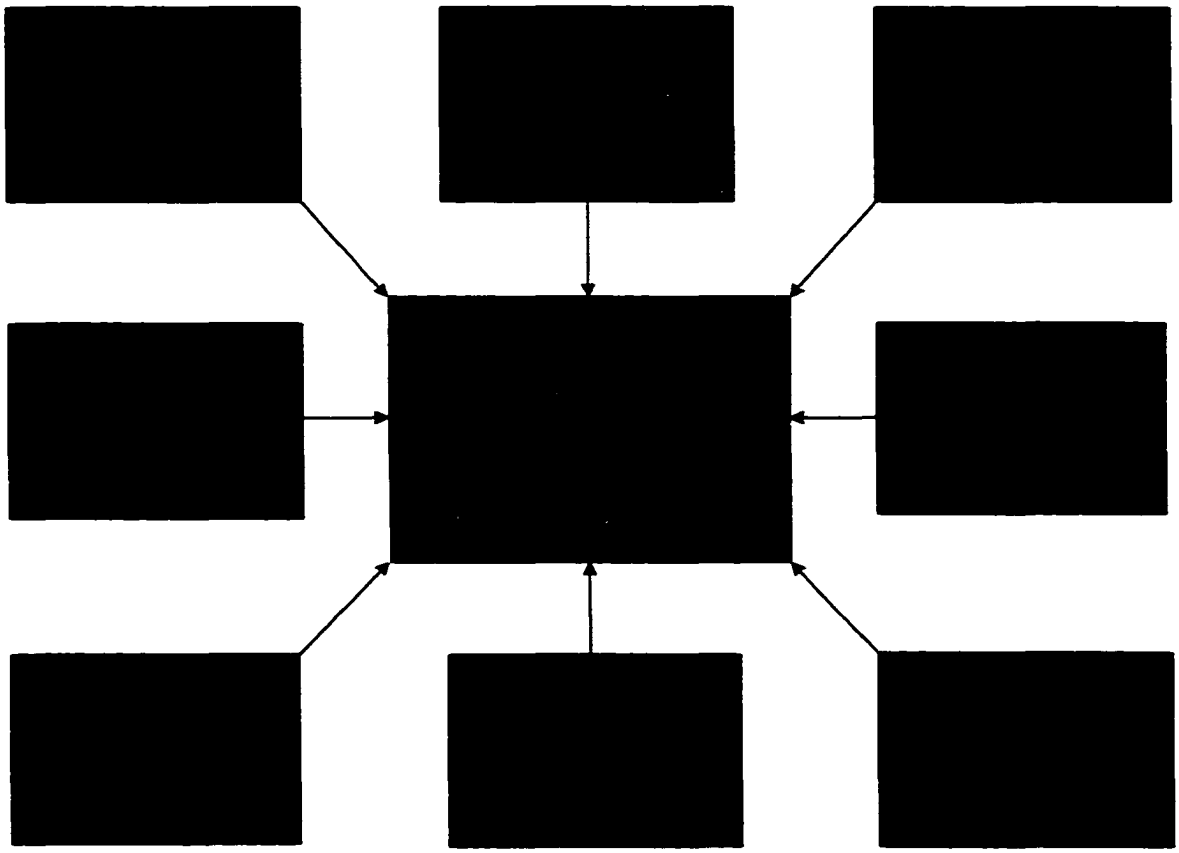


Figure 80 : Synthèse du filtre de Kirsch

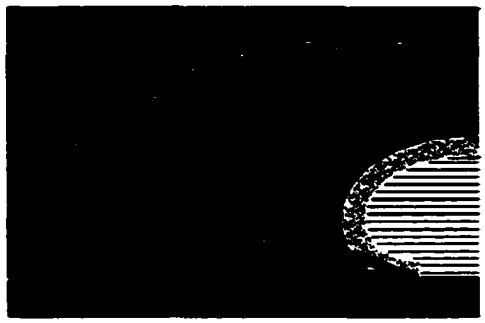


Figure 81 : Segmentation de la figure 80

Figure 82 : Les deux ouvertures appliquées à la figure 81

7.3 Carte des discontinuités verticales

Trouver les discontinuités verticales dans l'environnement est un processus très semblable à la recherche des pentes. On utilise la dérivée seconde pour identifier le passage par zéro de la dérivée première de l'image. On approxime la dérivée seconde par le calcul du laplacien. L'opérateur de dérivée seconde est celui-ci :

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2}\mathbf{i} + \frac{\partial^2 I}{\partial y^2}\mathbf{j} \tag{7.9}$$

où : $\nabla^2 I$ est la dérivée seconde de l'image

i et *j* indiquent les éléments du vecteur directeur

Une propriété importante de cette dérivée est la suivante : elle est proportionnelle à la hauteur de la discontinuité verticale (Δy sur la figure 66).

$$\nabla^2 I \propto \Delta y$$

Encore une fois, on utilise l'opérateur de convolution (équation 7.2) avec un noyau de type laplacien pour approximer l'opérateur de dérivée seconde.

$k_1 = \frac{1}{4}$

0	-1	0
-1	4	-1
0	-1	0

Figure 83 : Filtre laplacien

L'évaluation de la carte des discontinuités verticales se termine par un filtre médian éliminant les résidus ponctuels et par une fermeture utilisant un élément structurant de grande taille. Cet opérateur morphologique permet de fermer les régions mal segmentées et de limiter le déplacement du robot près des obstacles rapprochés. Cette fermeture utilise un élément structurant proportionnel à la taille du robot dans la RVE.

Les figures suivantes montrent le résultat des différentes opérations utilisant deux seuils de Δy_{max}. On peut voir les endroits correspondant aux discontinuités verticales trop grandes. Les régions blanches peuvent être interprétées comme des obstacles infranchissables pour le robot. On remarque que, lorsque Δy est grand, les escaliers ne sont pas considérés comme un obstacle. Par contre, lorsque Δy est petit, il ne reste qu'un chemin possible car la hauteur des contremarches est considérée comme un obstacle.

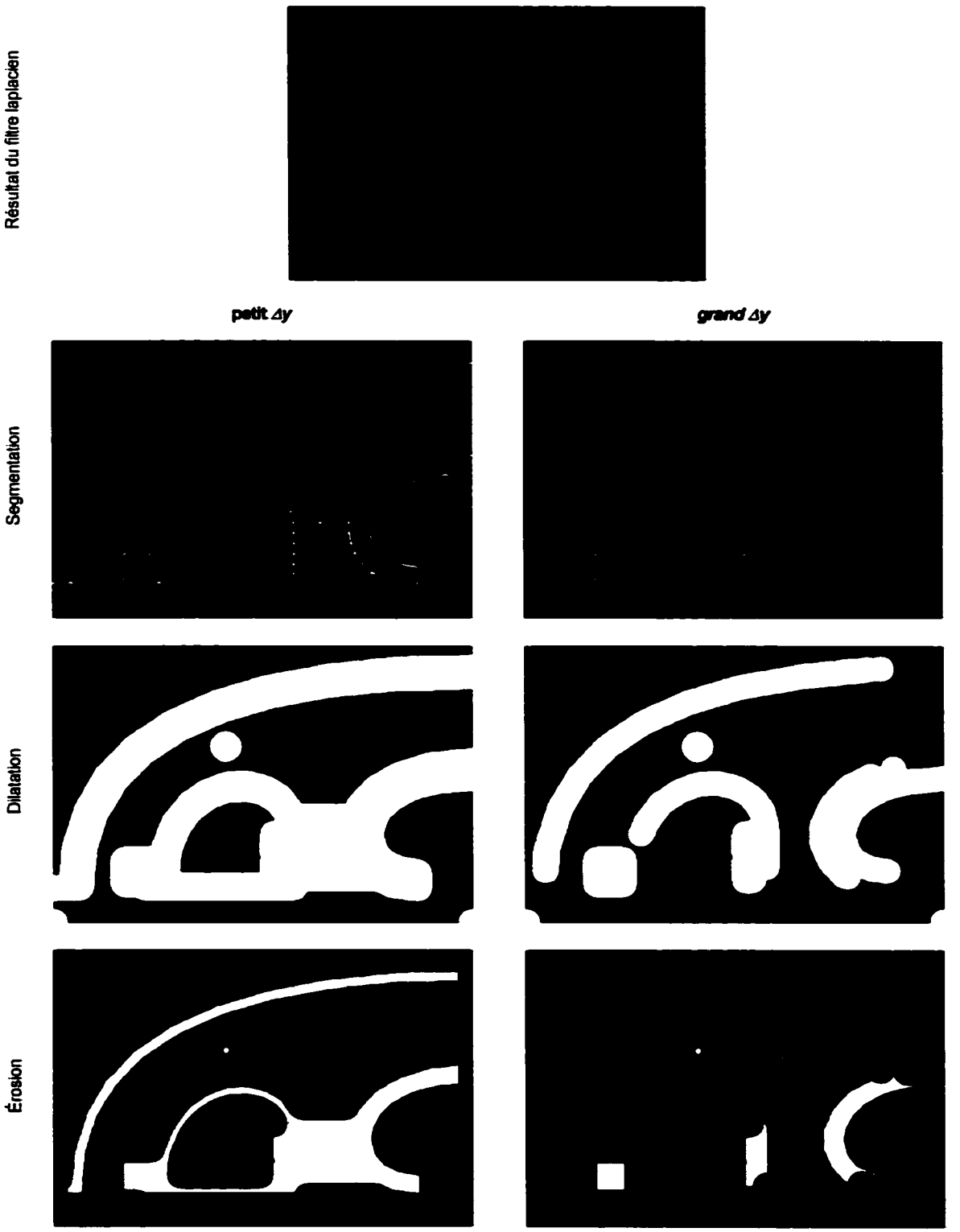


Figure 84 : Processus d'évaluation des discontinuités verticales

7.4 Synthèse des différentes représentations

Puisque chacune des cartes indique des obstacles de nature différente, il faut les considérer simultanément dans la carte finale.

$$C = CP_{el} \vee CD \quad \text{ou} \quad C = CP_{m} \vee CD \tag{7.10}$$

où : C est la carte finale
CD est la carte des discontinuités verticales
CP_{el} est la carte des pentes présentée par la technique de l'espace libre
CP_{m} est la carte des pentes présentée par la technique des pentes maximums
$\vee$ est l'opérateur correspondant à un *ou logique* entre les images

Pour la représentation des pentes par l'espace libre, on projette CD sur le plan x-y de CP_{el} traversant le volume le long de l'axe θ.

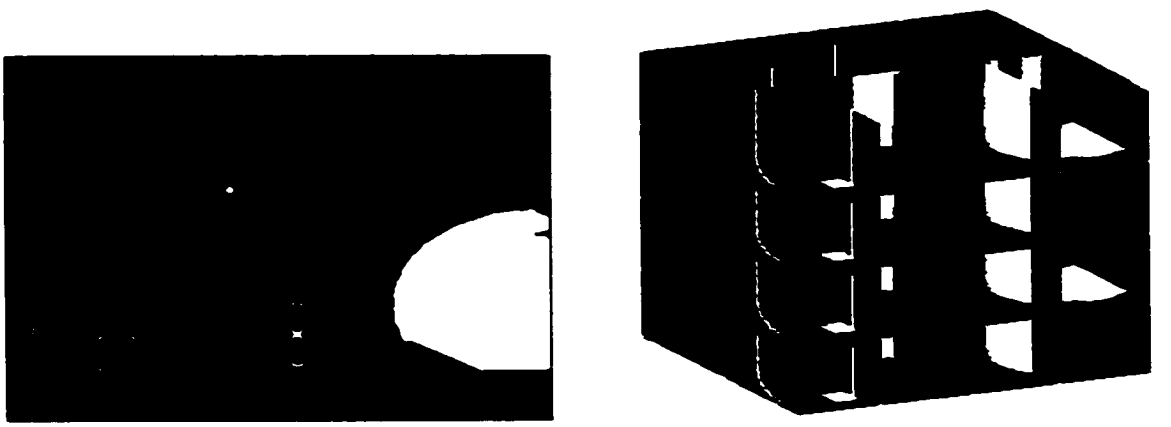


Figure 85 : Cartes virtuelles

CONCLUSION ET RECOMMANDATIONS

L'objectif principal de cette deuxième partie du mémoire est de développer une technique permettant d'interfacer les modules de perception de l'environnement et de planification de trajectoire. La représentation complexe et éparse obtenue par le nuage de points ne permet pas une compréhension suffisante de l'environnement pour réussir une planification de trajectoire.

L'approche présentée consiste à identifier l'endroit où se trouvent les obstacles sur le terrain. On commence par créer une représentation de type surface d'élévation à partir du nuage de points. Ensuite, on évalue les endroits du terrain correspondant aux deux types d'obstacles considérés : les pentes et les discontinuités verticales. On montre qu'avec les opérateurs de dérivées première et seconde, il est possible d'identifier ces types d'obstacles. L'utilisation de filtres morphologiques de type ouverture et fermeture permet de localiser les endroits spécifiquement intéressants selon les critères définis.

De plus, on propose un nouveau noyau de convolution permettant d'évaluer la dérivée première d'une image. L'avantage de ce noyau de convolution est de calculer l'amplitude de la dérivée première en tout point de l'image pour une orientation spécifique par une simple convolution.

On propose trois recommandations pour cette partie.

a. Il importe de trouver une méthode pouvant compresser les données. La somme colossale de données rend cette représentation trop lourde dans le cas où le robot doit parcourir une grande distance.

b. Il est possible d'améliorer grandement le modèle représentant l'environnement en considérant les incertitudes de mesure. En tenant compte des erreurs de mesure, on peut implanter une méthode de planification de trajectoire permettant de minimiser

les risques de collision. Il ne faut pas perdre de vue que plusieurs techniques de positionnement du robot par rapport à l'environnement ont une dérive importante dans le temps. Voilà pourquoi tenir compte de toutes les incertitudes peut permettre d'améliorer la qualité de la solution globale.

c. Finalement, il est essentiel d'implanter une technique pouvant détecter les crevasses afin de profiter de toutes les possibilités offertes par le robot marcheur.

TROISIÈME PARTIE

PLANIFICATION DE TRAJECTOIRE

INTRODUCTION

L'objectif du module de planification de trajectoire est de définir une trajectoire viable permettant à un corps physique de se déplacer d'un point A à un point B sur un terrain contenant divers obstacles (de nature et de formes différentes). Les critères d'optimisation de la trajectoire peuvent varier en fonction de l'application et sont limités par l'approche algorithmique utilisée pour résoudre le problème. Le critère le plus couramment utilisé est celui-ci : trouver le chemin offrant la distance la plus courte. D'un autre côté, certains algorithmes de planification permettent de déterminer le chemin le plus rapide et d'offrir ainsi le temps de parcours le plus court en considérant les contraintes cinématiques du robot, ce qui est souvent préférable. Aussi, lorsque le terrain est un environnement extérieur non structuré, il est intéressant de considérer le chemin le moins exigeant énergétiquement. Un autre critère qui peut être considéré et qui minimise les situations précaires est de déterminer le chemin le plus court maximisant la distance du robot aux différents obstacles. Cette dernière approche, plus conservatrice quant à la trajectoire définie, est idéale pour les situations délicates. Elle permet de minimiser les collisions tout en assurant la réussite de l'objectif. Ce type de situation peut se présenter par exemple dans une usine où se trouvent des produits dangereux ou lors de l'exploration de régions éloignées et inconnues telles que les fonds marins ou jusqu'à la conquête de l'espace.

Il existe deux niveaux très différents de planification de trajectoire pour un robot marcheur autonome. Le premier niveau est la planification de la trajectoire du corps du robot pour se rendre à la position objective. Le deuxième niveau est la planification de trajectoire de chaque patte pour réussir à mouvoir le corps du robot de façon à suivre la trajectoire définie au premier niveau. Ce mémoire aborde seulement la problématique associée au premier niveau. Le lecteur intéressé par l'approche qui est développée pour le contrôle de la démarche d'un robot quadrupède dynamique peut se référer à Lessard (2002).

La recherche appliquée au problème de planification de trajectoire date déjà de plusieurs décennies. Par contre, ce n'est que depuis l'avènement des ordinateurs actuels que cette science a fait l'objet de recherches intenses. De plus, avec l'intérêt croissant pour les robots mobiles, le problème de planification de trajectoire devient crucial à résoudre. Ainsi plusieurs approches, technologies et solutions ont été développées, présentées, implantées et finalement évaluées. On présente maintenant plusieurs solutions intéressantes qui ont été investiguées.

La troisième partie du mémoire est divisée en trois chapitres. Le chapitre 8 présente une brève revue de littérature sur le sujet ainsi que l'approche retenue. Le chapitre 9 présente l'aspect algorithmique de cette approche. Ce chapitre présente l'approche du chemin le plus court suivant les contours d'obstacles polygonaux. Finalement, le chapitre 10 présente quelques simulations illustrant adéquatement toutes les étapes du projet (incluant la première et deuxième partie).

CHAPITRE 8

DÉFINITION DE LA PROBLÉMATIQUE

8.1 Revue de littérature

Avant de débuter cette revue de littérature, l'ouvrage de Latombe (1991) mérite d'être cité par la qualité de sa synthèse sur les techniques de bases en planification de trajectoire. C'est un ouvrage excellent pour ceux qui débutent dans le domaine et pour ceux qui désirent un livre de référence d'une rigueur exceptionnelle. D'ailleurs, cet ouvrage est l'une des plus grandes sources de références pour ce travail.

8.1.1 Espace libre

La RVE définie selon la méthode de l'espace libre présentée à la section 6.1.2 permet de trouver des solutions viables à des problèmes de planification de trajectoire complexes. La technique utilisée pour déterminer le chemin le plus court dépend grandement du type d'espace libre utilisé. Une technique souvent utilisée consiste à déterminer la trajectoire par évaluations semi-aléatoires successives. La figure suivante montre un exemple typique de cette méthode.

Cette solution présente l'avantage de trouver un chemin même dans les cas les plus complexes. Par contre, le temps requis pour le calcul est considérable et il est difficile de trouver le chemin respectant des critères d'optimisation. Lozano-Pérez (1983), Latombe (1991) et Pruski (1996) présentent de façon détaillée cette technique.

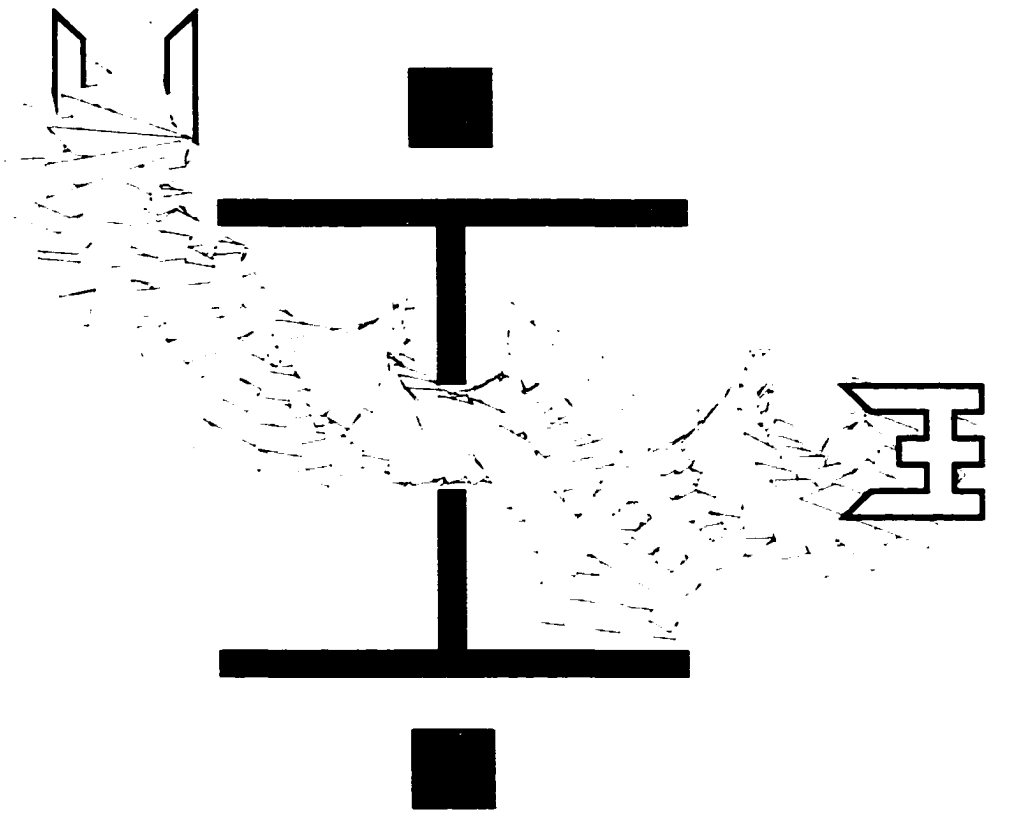


Figure 86 : Exemple de trajectoire définie de façon semi-aléatoire dans l'espace libre

8.1.2 Description polygonale des obstacles

La description polygonale des obstacles permet de simplifier la RVE de même que la planification de trajectoire. Cette technique géométrique est basée sur le fait suivant : le chemin le plus court est constitué des segments de droite passant par les sommets des obstacles. La figure suivante illustre un environnement où les obstacles sont définis par une description polygonale des contours. Le point de départ est le cercle blanc et le point d'arrivée est le cercle noir.

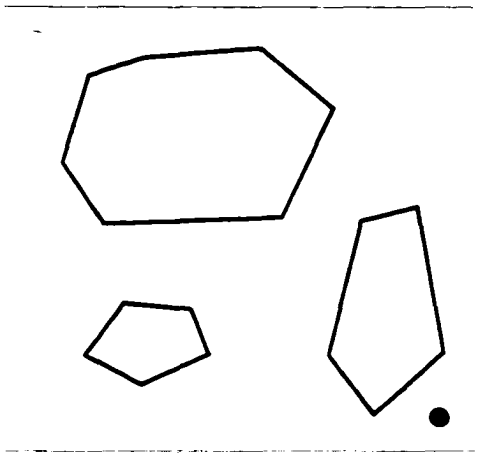
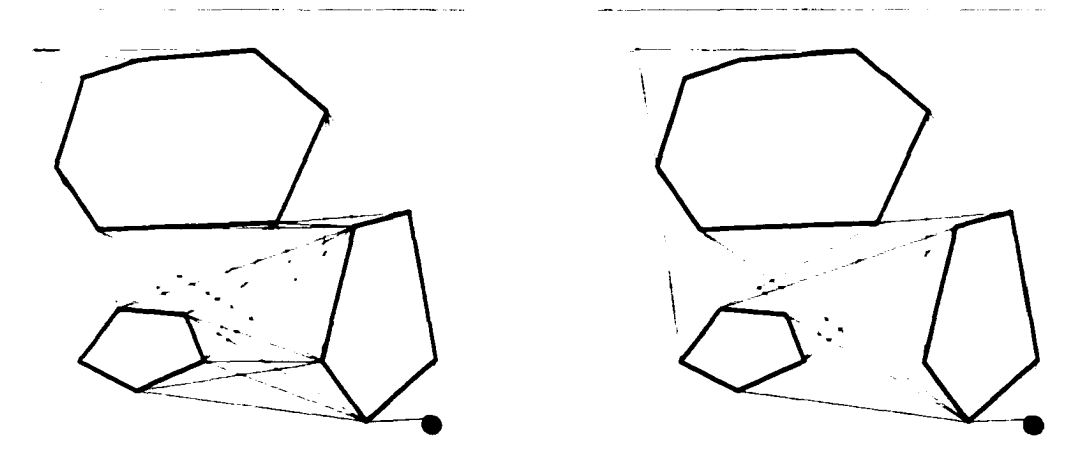


Figure 87 : Description polygonale des obstacles

On définit premièrement le graphe de visibilité qui représente toutes les connexions possibles pour chaque sommet sans entrer en collision avec un obstacle (figure 88a). Ces segments représentent tous les chemins possibles. À partir du graphe de visibilité, on trace le graphe de visibilité tangent (figure 88b). Ce graphe permet de restreindre le nombre de chemins aux chemins les plus courts contournant les obstacles.

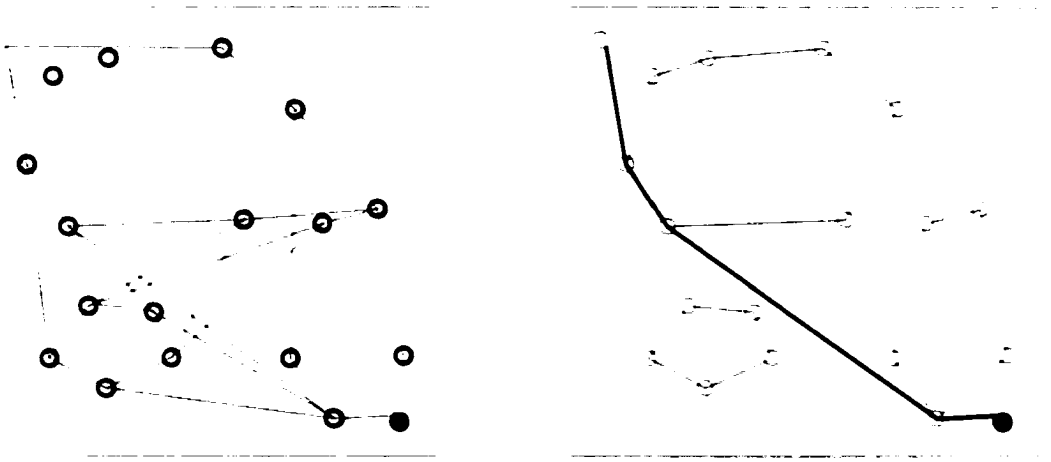


a) Graphe de visibilité

b) Graphe de visibilité tangent

Figure 88 : Graphes de visibilité

Tel que défini par les graphes de visibilité, on calcule la distance existant entre chaque segment du graphe (figure 89a). Finalement on trouve le chemin le plus court (figure 89b). L'algorithme de Dijkstra (1959) est certainement la méthode la plus utilisée pour résoudre le problème du chemin le plus court dans un graphe.

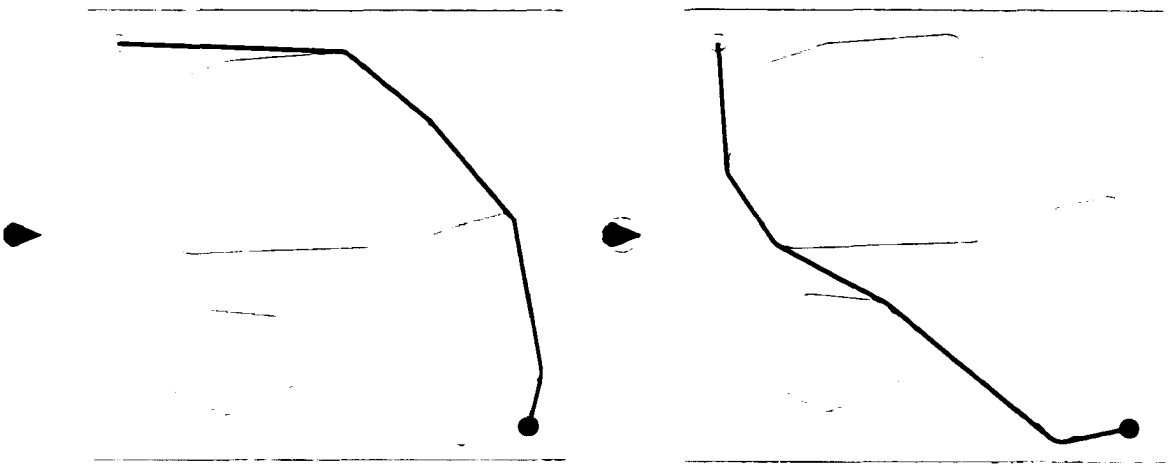


a) Graphe des chemins tangents b) Chemin le plus court

Figure 89 : Graphe des distances et chemin le plus court

Cette technique relativement simple est souvent utilisée mais aussi très limitée. Lorsque le nombre de sommets décrivant les obstacles augmente, le temps de calcul devient considérable. Aussi, cette approche ne considère pas les dimensions et les contraintes cinématiques du véhicule. En fait, on suppose que le véhicule est un point capable de changer de direction instantanément à chaque sommet.

Il existe évidemment des approches dérivées permettant de solutionner ces problèmes. Par exemple, on peut appliquer un algorithme de croissance des obstacles selon la dimension maximum du robot. Cette technique permet de définir un espace libre viable où le robot peut être considéré comme un point. Si on ne considère pas la rotation du mobile, on utilise sa description géométrique. Par contre, on utilise souvent un cercle dont le rayon correspond à la dimension maximale du mobile. La figure suivante illustre deux exemples de ces espaces libres où la forme du robot est celle-ci ◗.



a) Croissance des obstacles selon la géométrie du mobile

b) Croissance des obstacles par le cercle qui circonscrit le mobile

Figure 90 : Croissance des obstacles définissant l'espace libre

Une autre amélioration est apportée en considérant les contraintes cinématiques du robot. Si on considère un robot sur roue, les contraintes cinématiques sont simples à définir : elles correspondent à la vitesse et à l'angle maximum permis des roues directrices (rayon de courbure pour le rouleur $r_r = r_{max}$). Par contre, un robot sur pattes n'a pas cette contrainte puisque ce dernier peut s'immobiliser et tourner sur place pour s'orienter vers sa nouvelle destination (rayon de courbure pour le marcheur $r_m = 0$). Ces techniques ne sont pas très efficaces car elles ne considèrent pas la vitesse du robot. En effet, le rayon de courbure est fonction de la vitesse du robot et de la force de friction existante entre le robot et le sol. Donc, en considérant la vitesse, on obtient $r_r \geq r_{max}$ et $r_m > 0$. La figure suivante illustre une technique utilisée pour représenter les contraintes cinématiques lorsque le rayon de courbure est considéré constant, pour un robot à faible vitesse par exemple.

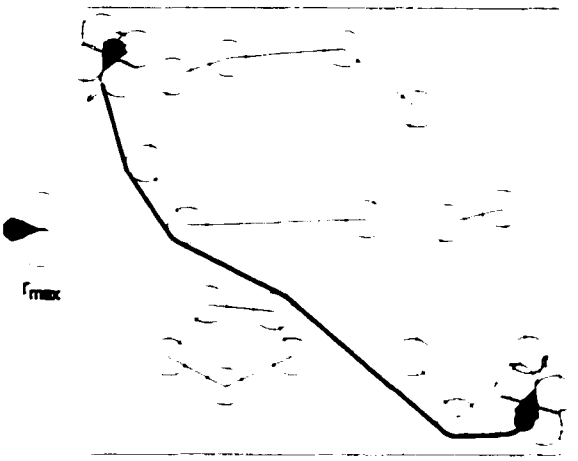


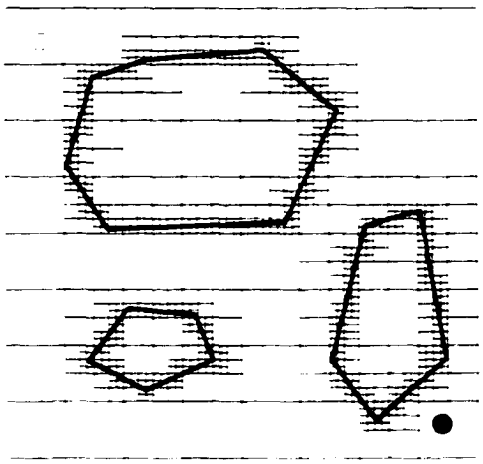
Figure 91 : Planification de trajectoire avec contrainte cinématique

La littérature est abondante sur ce sujet. À titre de référence le lecteur intéressé peut se référer à Whitesides (1985), Latombe (1991), Bicchi, Casalino et Santilli (1995), Pruski (1996), et à Agarwal, Biedl et Lazard (1998).

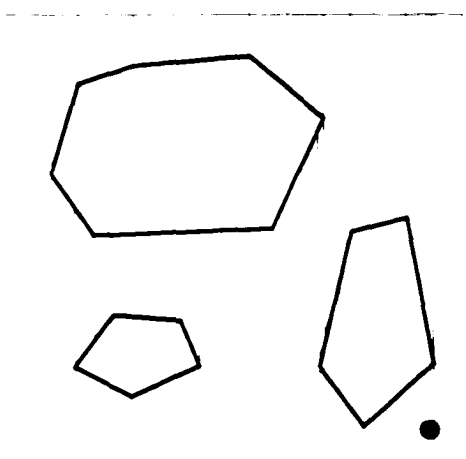
8.1.3 Décomposition cellulaire

L'approche par décomposition cellulaire permet de trouver un chemin selon les techniques de RVE présentées à la section 6.1.5. Il existe plusieurs techniques dont certaines sont plus élaborées. Les techniques les plus simples sont la recherche aléatoire et la recherche du chemin le plus court dans un graphe. Ce dernier est réalisé à partir des noeuds générés par chaque élément de la division cellulaire. Le chemin résultant peut être un des trois suivants : en passant par le centre des cellules, en longeant les frontières des cellules ou en parcourant le centre des frontières de chaque cellule. On utilise encore l'algorithme de Dijkstra pour calculer la distance la plus courte dans un graphe.

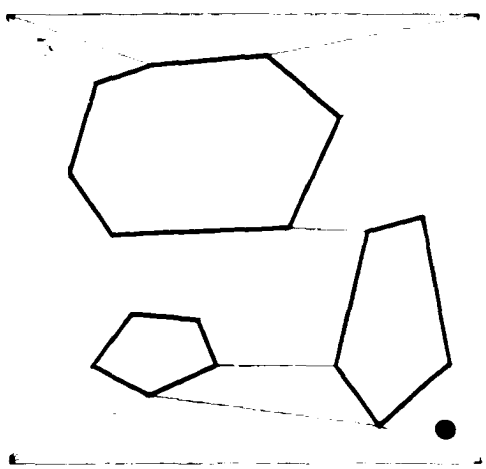
Les figures suivantes illustrent ces principes selon deux types de division cellulaire. La scène considérée est celle décrite à la figure 87.



a) Division cellulaire de type binaire

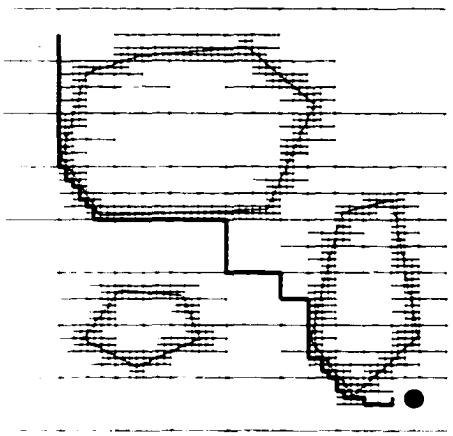


b) Division cellulaire trapézoïdale

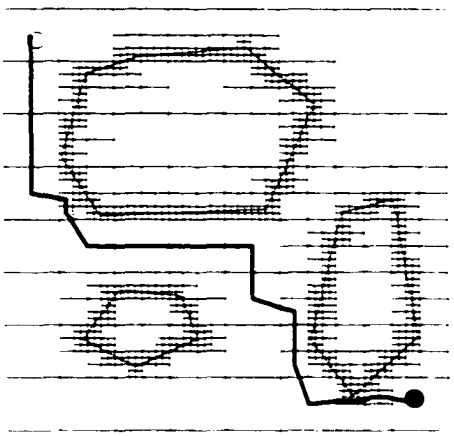


c) Division par polygones convexes

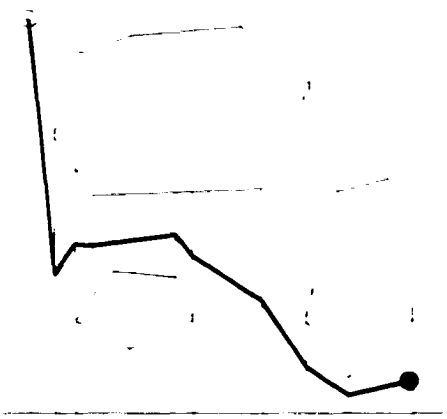
Figure 92 : Différents types de décomposition cellulaire



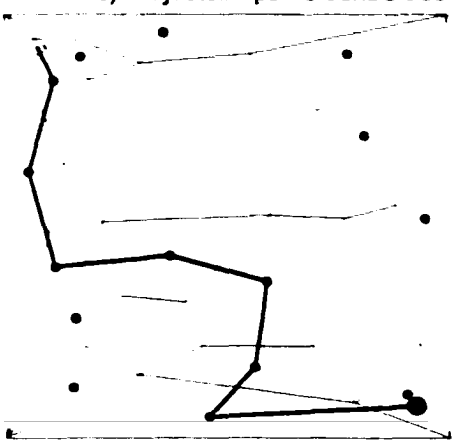
a) Trajectoire par les contours des cellules (figure 92a)



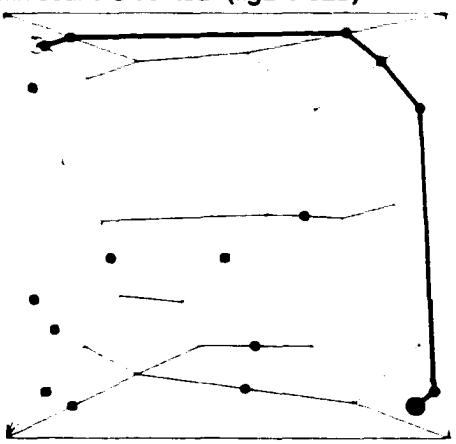
b) Trajectoire par les centres des cellules (figure 92a)



c) Trajectoire par le centre des segments définissant le contour (figure 92b)



d) Trajectoire par les centroïdes des polygones (figure 92c)



e) Trajectoire par les centres des frontières des polygones (figure 92c)

Figure 93 : Trajectoires sur les différents types de décomposition cellulaire

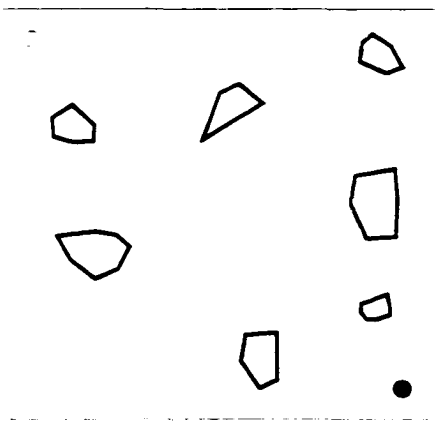
Encore une fois les différents aspects de cette technique sont abordés par Whitesides (1985). Latombe (1991) et Pruski (1996).

8.1.4 Maximisation de la distance aux obstacles

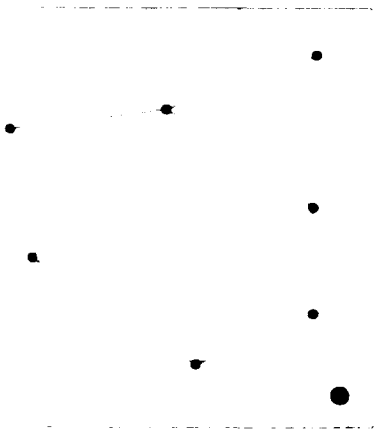
Cette approche consiste à maximiser la distance du robot par rapport aux obstacles. La section sur la RVE par la proximité des obstacles montre une représentation de l'espace libre par les endroits les plus favorables pour éviter les collisions. La technique la plus couramment utilisée pour solutionner le problème de planification de trajectoire selon cette approche est basée sur le diagramme de Voronoi. Cette technique consiste à segmenter l'espace par des droites et des segments de paraboles indiquant les chemins qui maximisent la distance aux obstacles.

Il existe plusieurs façons de faire le diagramme de Voronoi. La plus simple se construit de la façon suivante : on applique l'algorithme de triangulation de Delaunay sur la représentation ponctuelle des obstacles. Chaque noeud indique un obstacle et chaque segment connecte les obstacles les plus près entre eux. On construit ensuite les chemins les plus distants des obstacles par l'intersection des droites médianes perpendiculaires aux différents segments issus de la triangulation de Delaunay. Les figures suivantes illustrent cette approche.

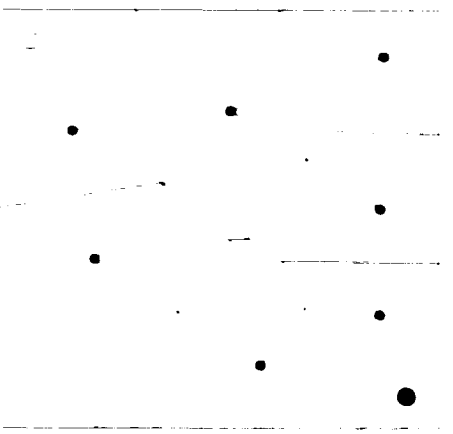
Ce problème de nature géométrique est bien documenté dans les ouvrages traitant de géométrie. À titre de référence, on cite les ouvrages de Preparata et Shamos (1985), Boissonnat et Yvinec (1998) ainsi que Berg, van Kreveld, Overmars et Schwarzkopf (1998). Les ouvrages de Alt et Yap (1990), Latombe (1991) et Pruski (1996) abordent le problème du point de vue de la planification de trajectoire.



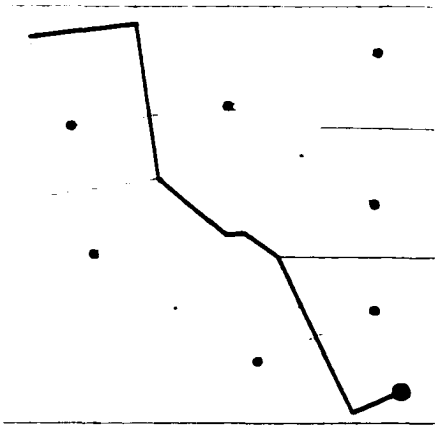
a) Scène prise pour l'exemple



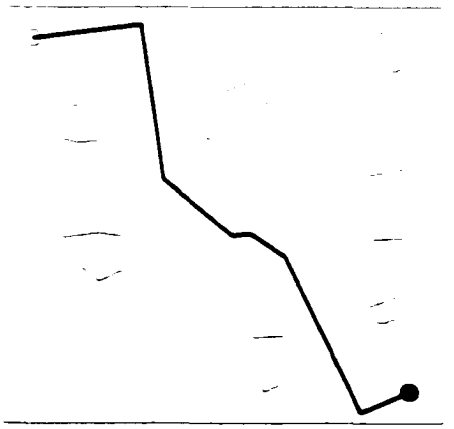
b) Triangulation de Delaunay sur la représentation ponctuelle des obstacles



c) Diagramme de Voronoi



d) Chemin le plus court à l'aide du diagramme de Voronoi



e) Chemin le plus court sur la scène

Figure 94 : Chemin le plus court à l'aide du diagramme de Voronoi

8.1.5 Champ de potentiel artificiel

Cette approche consiste à générer artificiellement un champ de force attirant le mobile vers la position objective. On génère deux champs opposés : un champ attractif attirant le mobile vers la destination et un champ répulsif repoussant le mobile loin des obstacles. La solution simple est facile à réaliser mais produit des minimums locaux et nécessite des algorithmes externes pour contourner ou sortir de ces minimums.

Voici une définition simple du champ de potentiel artificiel.

$$\Phi_{x,y} = U_{x,y} + V_{x,y} \tag{8.1}$$

où : $U_{x,y}$ est le champ de potentiel attractif
$V_{x,y}$ est le champ de potentiel répulsif

$$U_{x,y} = \frac{1}{2} k_0 \left(\sqrt{(x-x_o)^2 + (y-y_o)^2} \right)^2 \equiv \frac{1}{2} k_0 \left((x-x_o)^2 + (y-y_o)^2 \right) \tag{8.2}$$

$$V_{x,y} = \begin{cases} \frac{1}{2} \cdot \eta \cdot \left(\frac{1}{\rho} - \frac{1}{\rho_0} \right)^2 & \text{si } \rho \le \rho_0 \\ 0 & \text{si } \rho > \rho_0 \end{cases} \tag{8.3}$$

où : (x_o, y_o) est la coordonnée de l'objectif
k_0 est une constante définissant l'amplitude du champ attractif
η est une constante définissant l'amplitude des obstacles sur le champ répulsif
ρ est la distance entre (x, y) et l'obstacle le plus près
ρ_0 est la distance limite de l'influence du champ de potentiel répulsif

On calcule ensuite le gradient sur l'image pour connaître en tout point la direction vers laquelle se diriger. L'orientation et l'amplitude du gradient se calculent par les

équations 7.3 et 7.4 en utilisant les noyaux de convolution désirés. Dans l'exemple suivant, les deux noyaux de Sobel sont utilisés.

La scène suivante sert à illustrer l'algorithme.

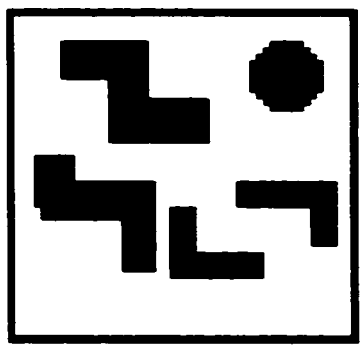
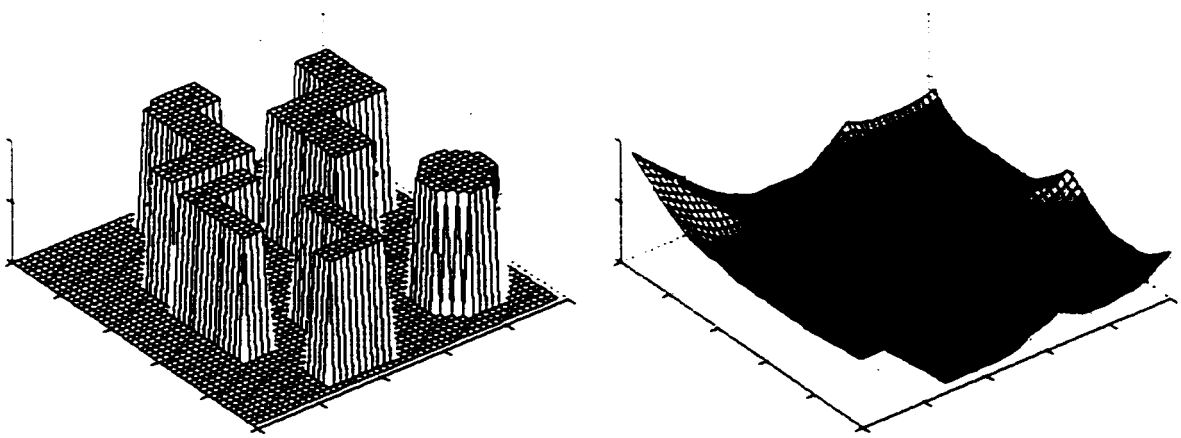


Figure 95 : Scène utilisée pour illustrer la méthode de champ de potentiel artificiel

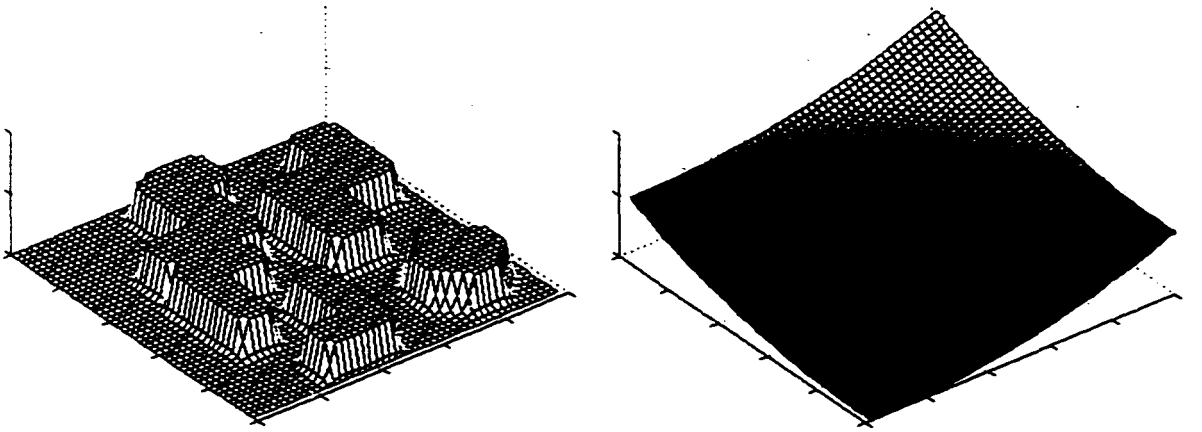
On commence par calculer le champ répulsif à l'aide de la carte des distances préalablement calculée (telle qu'illustrée à la section 6.1.8). On calcule ensuite les champs attractif et répulsif pour finalement évaluer le champ de potentiel artificiel total.



a) Surface d'élévation de la scène

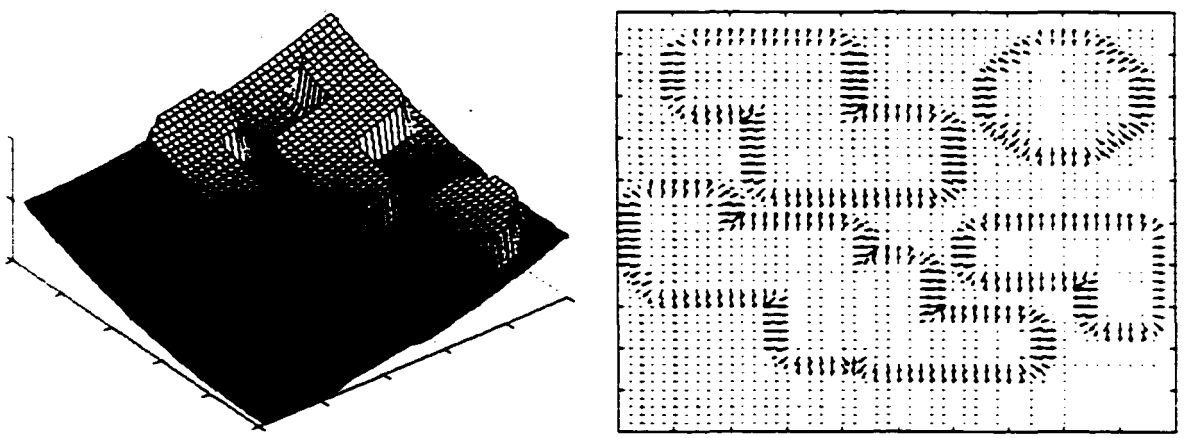
b) Carte des distances

Figure 96 : Calcul de la carte des distances



a) Champ répulsif b) Champ attractif

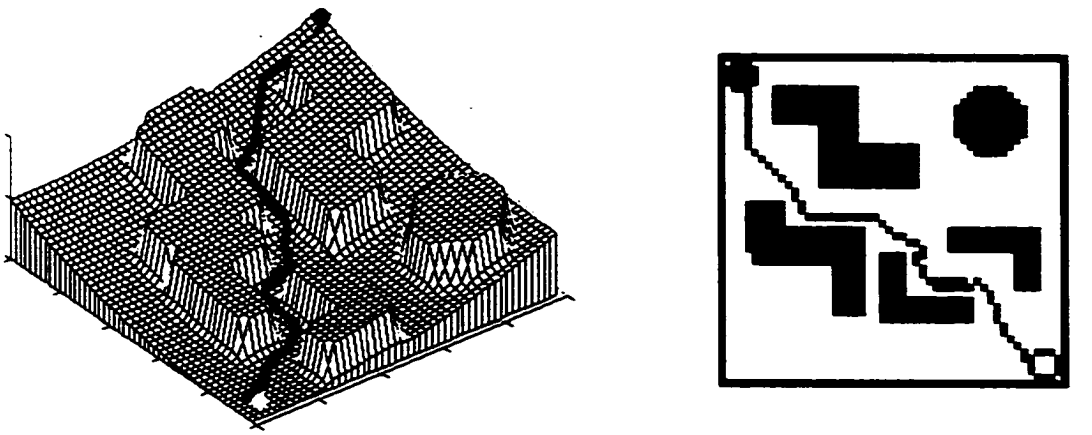
Figure 97 : Calcul des champs répulsif et attractif



a) Champ de potentiel artificiel b) Gradient du champ de potentiel artificiel

Figure 98 : Calcul du champ de potentiel artificiel et du gradient

Finalement, on évalue la trajectoire.



a) Trajectoire
(vue sur le champ de potentiel artificiel)

b) Trajectoire
(vue sur la scène)

Figure 99 : Trajectoire obtenue par la technique du champ de potentiel artificiel

En changeant les coordonnées de départ et d'arrivé, le champ de potentiel artificiel est différent et il est possible que la technique présentée provoque des minimums locaux selon la forme des obstacles. On reprend l'exemple précédent de façon telle qu'on provoque le passage par la concavité d'un obstacle. On voit que la méthode de suivi de gradient est inadéquate et ne permet pas de quitter le minimum.

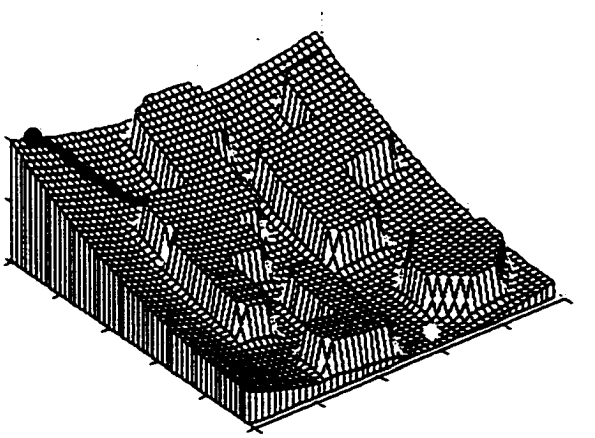


Figure 100 : Exemple d'un minimum local avec la méthode du champ de potentiel artificiel

Plusieurs techniques permettent soit de quitter les minimums, soit d'éviter ces minimums ou soit tout simplement de ne pas créer de minimum. Latombe (1991) présente en détail plusieurs alternatives et variations sur le même thème. Une technique plus récente très performante consiste à utiliser un champ de potentiel artificiel harmonique. Avec cette approche, aucun minimum n'est créé (voir Magherbi et Wolovich (1992), Guldner et Utkin (1993) et Mantegh, Jenkin et Goldenberg (1997)).

8.1.6 Autres approches de planification de trajectoire

Les méthodes qui viennent d'être énumérées sont parmi les plus utilisées autant par leur simplicité que par les succès obtenus. D'un autre côté, le sujet de la planification de trajectoire est vraiment très vaste et il est impossible de présenter complètement ce sujet par une aussi brève revue de littérature. Voici néanmoins d'autres approches offrant des alternatives intéressantes et prometteuses.

Cunha et Coimbra de Matos (1994) proposent une formulation stochastique du problème. Cette solution considère les critères de consommation énergétique, du risque de collision et du temps de déplacement. Les principes de la programmation dynamique sont utilisés pour résoudre ce problème.

Dans le cadre d'une application réelle, les incertitudes de mesures ou de positionnement du robot dans l'espace posent des problèmes difficiles à résoudre. On utilise alors des points de repère pour évaluer la position réelle du robot et ainsi diminuer l'intervalle d'incertitude. Lazanas et Latombe (1995) montrent une approche basée sur des régions connues de l'espace où le positionnement et le comportement du robot sont stables. Cette approche permet une navigation complète et donne des résultats très intéressants. D'un autre côté, Howard et Kitchen (1998) montrent comment arriver au but sans même connaître le point de départ du robot. Ces deux références en sont deux parmi tant d'autres car la littérature abonde sur ce sujet.

Plusieurs méthodes sont présentées pour solutionner le problème de planification de trajectoire sur un terrain accidenté. Ces méthodes sont souvent basées sur la modélisation dynamique du véhicule, la topographie du terrain, les obstacles et les éléments de surface. Elles donnent une trajectoire définie en position et en vitesse. On cite à titre d'exemple Shiller et Gwo (1991), Siméon et Dacre-Wright (1993), Cherif (1999), Shiller (1999) et Donald, Xavier, Canny et Reif. Une autre approche donnant des résultats très intéressants est présentée par Pai et Reissell (1998). Leur approche consiste en une modélisation multirésolution du terrain sous forme d'ondelettes (*wavelets*). Pagnot et Grandjean (1995) proposent à leur tour une méthode très simple et rapide de planification de trajectoire immédiate basée sur le balayage vertical du sol devant le robot.

Les algorithmes génétiques sont la cible d'intérêt de nombreux chercheurs depuis quelques années. Cette approche d'optimisation est aussi adaptée pour le problème de planification de trajectoire. Ashiru et Czarnecki (1995) démontrent la validité de l'approche pour un environnement non structuré et incertain. Sugihara et Smith (1997) montrent comment les algorithmes génétiques sont puissants pour solutionner le problème lorsqu'on considère une grande adaptabilité sur le terrain. Farritor et Dubowsky (1997) utilisent les algorithmes génétiques pour déterminer une trajectoire en terrain accidenté tout en minimisant les risques de défaillance pour les robots explorateurs.

8.2 Approches retenues

La planification de trajectoire est un problème complexe. Il est difficile d'implanter une méthode répondant à tous les critères d'optimisation désirés et nécessitant un temps de calcul raisonnable. En fait, les méthodes de planification de trajectoire sont extrêmement exigeantes, ainsi il importe d'optimiser la structure de données et les algorithmes. De plus, avec les outils de calcul actuellement disponibles, il est essentiel

d'utiliser des méthodes non exhaustives permettant une approximation intéressante de la solution optimale.

Au moment d'écrire ces lignes, aucun algorithme de planification de trajectoire n'est sélectionné de façon définitive pour le projet Capra. Seulement deux méthodes sont explorées pour valider leur utilisation et leur performance : la méthode par champ de potentiel artificiel et la méthode conventionnelle basée sur la description polygonale des obstacles.

La méthode par champ de potentiel artificiel est succinctement implantée et aucune des méthodes permettant d'éviter les minimums n'est explorée encore. Sauf l'aspect algorithmique du problème, l'implantation actuellement réalisée se résume à ce qui a été présenté à la section 8.1.5. Par contre, l'autre méthode est explorée plus à fond et mérite d'être décrite plus en détail. Le chapitre 9 présente la solution algorithmique de la méthode du chemin le plus court suivant les contours d'obstacles polygonaux.

CHAPITRE 9

PLANIFICATION DE TRAJECTOIRE SUIVANT LES CONTOURS D'OBSTACLES POLYGONAUX

Cette méthode de planification de trajectoire est l'une des plus simples et offre une solution optimale en terme de distance parcourue. Par contre, elle ignore les contraintes cinématiques du robot et certains autres critères comme le temps de parcours. Cette méthode est basée sur une description polygonale du robot et des obstacles (section 6.1.6). Pour symboliser le robot dans l'espace libre par un point, on redimensionne les obstacles en fonction de la forme du robot. De cette description polygonale, on trace le graphe de visibilité qui indique tous les chemins possibles (voir la section 8.1.2). Finalement, on sélectionne le chemin le plus court. La figure suivante illustre tout le processus algorithmique de cette méthode de planification de trajectoire.

Il existe plusieurs variantes du processus algorithmique global ainsi que de chaque sous-algorithme. On présente ici de façon détaillée quelques solutions traitant de certains aspects du problème.

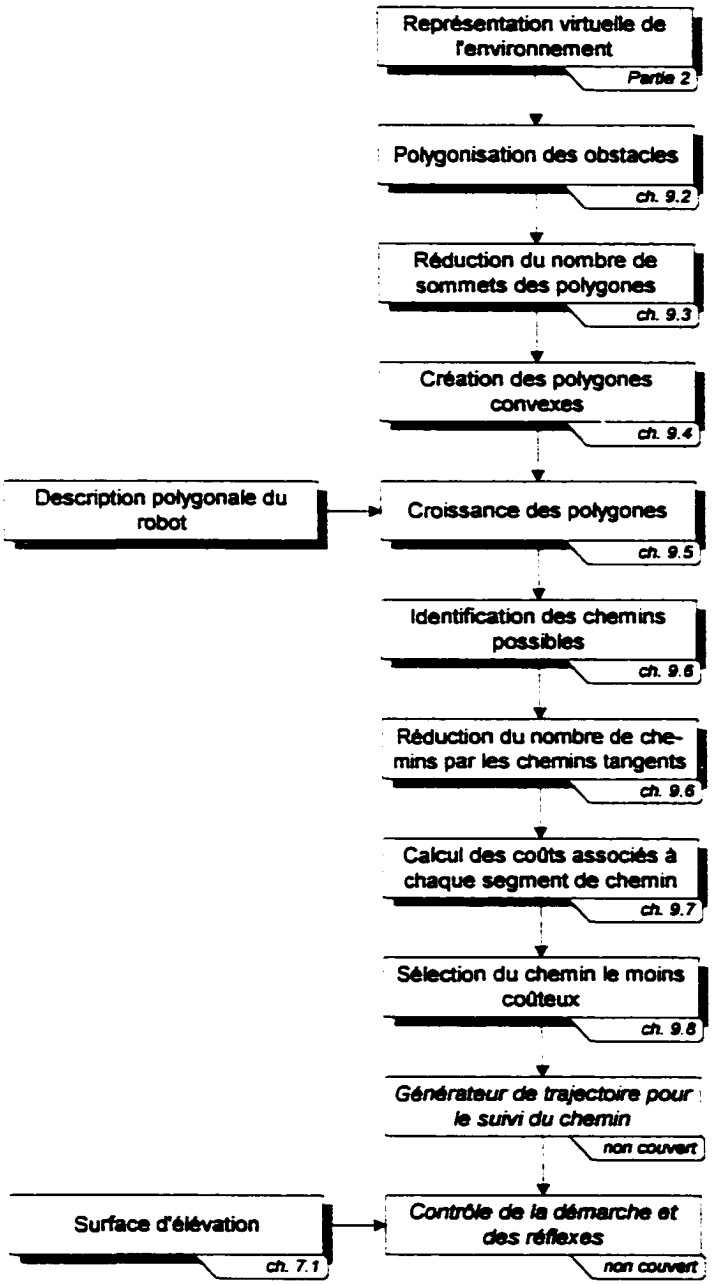


Figure 101 : Processus algorithmique de la planification de trajectoire suivant les contours des obstacles polygonaux

9.1 Concept des vecteurs traversables

Avant de commencer la description des algorithmes de cette approche, il est pertinent d'introduire dès maintenant la notion de vecteurs traversables. Cette notion permet de résoudre de façon élégante les problèmes reliés aux collisions entre un polygone et un point, un segment de droite ou un autre polygone. Plusieurs algorithmes décrits dans cette section utilisent ces notions. Pour cette raison, on présente cette méthode simple et efficace permettant de tirer plusieurs informations sur les polygones. Le concept des vecteurs traversables (*traversability vectors*) a été présenté par Janet, Luo et Kay (1997) et repose sur les demi-plans créés par les différents segments de droites des polygones. Cette section présente une brève revue des concepts fondamentaux des vecteurs traversables. Le lecteur peut se référer aux auteurs déjà cités pour prendre connaissance des différentes preuves et des possibilités offertes par les vecteurs traversables. Il est important de noter que la théorie associée aux vecteurs traversables n'est valable que pour les polygones convexes (la section 9.4 présente la définition d'un polygone convexe et montre comment les créer).

On pose P, un polygone constitué de n sommets. En calculant les demi-plans associés aux différents segments du polygone, on peut déterminer rapidement si un point u se trouve à l'intérieur ou à l'extérieur du polygone. Les demi-plans sont calculés pour tous les segments définis selon le même sens de rotation autour du polygone. On calcule le modèle des demi-plans Q du polygone P par :

$$Q = \overline{A} \cdot u - \overline{c} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \cdots & \cdots \\ a_{n1} & a_{n2} \end{bmatrix} \cdot \begin{bmatrix} u_x \\ u_y \end{bmatrix} - \begin{bmatrix} c_1 \\ c_2 \\ \cdots \\ c_n \end{bmatrix} \leq 0 \qquad (9.1)$$

où : $a_{i1} = u_{y_i} - u_{y_{i-1}}$

$a_{i2} = u_{x_{i-1}} - u_{x_i}$

$c_i = u_{x_{i-1}} \cdot u_{y_i} - u_{x_i} \cdot u_{y_{i-1}}$

Q est un vecteur binaire indiquant où se trouve u par rapport à P. La figure suivante montre un exemple des régions pour lesquelles Q est identique. On remarque que chaque élément du vecteur Q est égal à 1 lorsqu'un point est à l'intérieur de P.

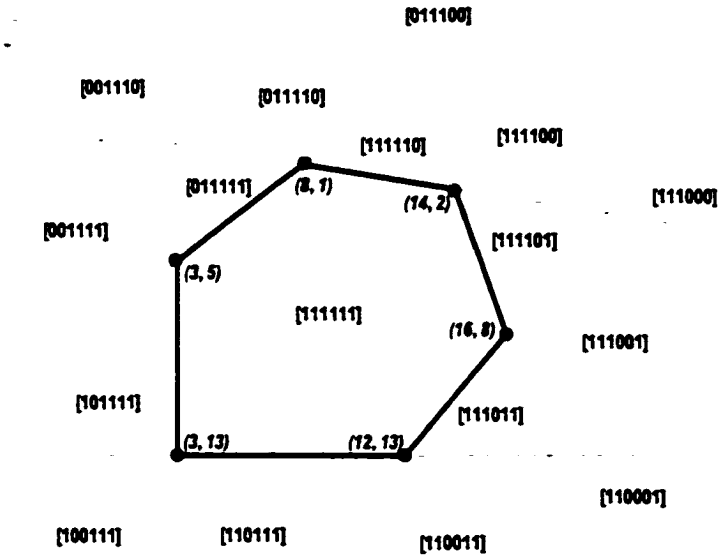


Figure 102 : Valeur du vecteur binaire Q pour différentes régions autour d'un polygone convexe

On peut maintenant connaître rapidement si un point est à l'intérieur ou à l'extérieur d'un polygone. Pour déterminer si un segment de droite entre en collision avec un polygone, on procède par deux étapes. La première étape consiste à déterminer si le segment $\overline{u_i u_j}$ est potentiellement obstrué par le polygone. La deuxième étape permet

de lever l'ambiguïté lorsque la première étape ne permet pas de déterminer si le segment est définitivement obstrué.

Pour évaluer si un segment de droite est potentiellement obstrué par un polygone, on applique l'opérateur logique *ou bit à bit* aux deux vecteurs binaires Q_1 *et* Q_2 des deux sommets du segment de droite. Si le vecteur résultant est constitué seulement d'éléments égaux à 1, alors on peut dire que le segment est potentiellement obstrué par le polygone et on doit résoudre le problème à l'aide de la deuxième étape. Par contre, si un ou plus d'un élément du vecteur résultant est 0, on peut affirmer que le segment n'entre pas en collision avec le polygone. La figure suivante montre trois segments de droites qui illustrent les différentes situations.

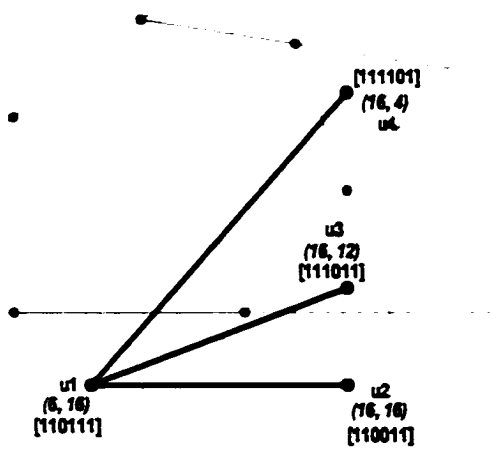


Figure 103 : Comment déterminer si un segment entre en collision avec un polygone convexe

Pour cette figure, on a les résultats suivants :

- $\overline{u_1u_2}$ donne $Q_1 \vee Q_2 = [110111] \vee [110011] = [110111] \Rightarrow$ aucune collision
- $\overline{u_1u_3}$ donne $Q_1 \vee Q_3 = [110111] \vee [111011] = [111111] \Rightarrow$ collision potentielle
- $\overline{u_1u_4}$ donne $Q_1 \vee Q_4 = [110111] \vee [111101] = [111111] \Rightarrow$ collision potentielle

La deuxième étape permettant de lever l'incertitude consiste à vérifier si tous les sommets du polygone se trouvent dans le même demi-plan généré par le segment. On calcule maintenant le vecteur D de cette façon :

$$D_{\overline{u_i u_k}} = \overline{A} \cdot u - \overline{c} = [a_{11} \quad a_{12}] \cdot \begin{bmatrix} P_{i_x} \\ P_{i_y} \end{bmatrix}_i - [c] \tag{9.2}$$

où : $a_{11} = u_{j_y} - u_{k_y}$

$a_{12} = u_{k_x} - u_{j_x}$

$c = u_{j_y} \cdot u_{k_x} - u_{j_x} \cdot u_{k_y}$

Lorsqu'il y a collision, on vérifie si tous les points se situent sur le même demi-plan. Pour vérifier cette condition, on s'assure que tous les éléments du vecteur D sont de la même polarité. Les éléments du vecteur D ayant une valeur de 0 indiquent que D passe par un sommet du polygone. Il est donc possible de gérer ces cas selon le type d'application.

En prenant les segments $\overline{u_1u_3}$ et $\overline{u_1u_4}$ de la figure 103 on obtient :

- $D_{\overline{u_1u_3}} = [-142 \quad -122 \quad -42 \quad -6 \quad -40 \quad -108] \Rightarrow$ aucune collision
- $D_{\overline{u_1u_4}} = [-126 \quad -146 \quad -66 \quad 42 \quad 40 \quad -44] \Rightarrow$ une collision

On peut étendre ces principes pour maintenant évaluer si un polygone entre en collision avec un autre. Il suffit de parcourir tous les segments du premier polygone et de vérifier s'ils entrent en collision avec le deuxième polygone. De plus, la technique des vecteurs traversables permet de déterminer beaucoup d'autres caractéristiques comme : connaître les sommets d'un polygone les plus près d'un point, identifier les vecteurs tangents entre deux polygones et savoir si un sommet se trouve devant ou derrière un polygone.

9.2 Polygonisation des obstacles

À l'entrée du processus de planification de trajectoire se trouve une RVE binaire représentant les obstacles de la scène. La tâche de polygonisation consiste à représenter sous forme de polygones les régions contiguës représentant les obstacles. Pour réaliser cette tâche, on utilise une méthode semblable à celle utilisée pour le code de Freeman (voir la section 4.1.1.1). On identifie le premier point d'un obstacle et ensuite on décrit son contour par une suite d'informations décrivant chaque élément par :

- la direction de l'élément voisin;
- la coordonnée de chaque élément.

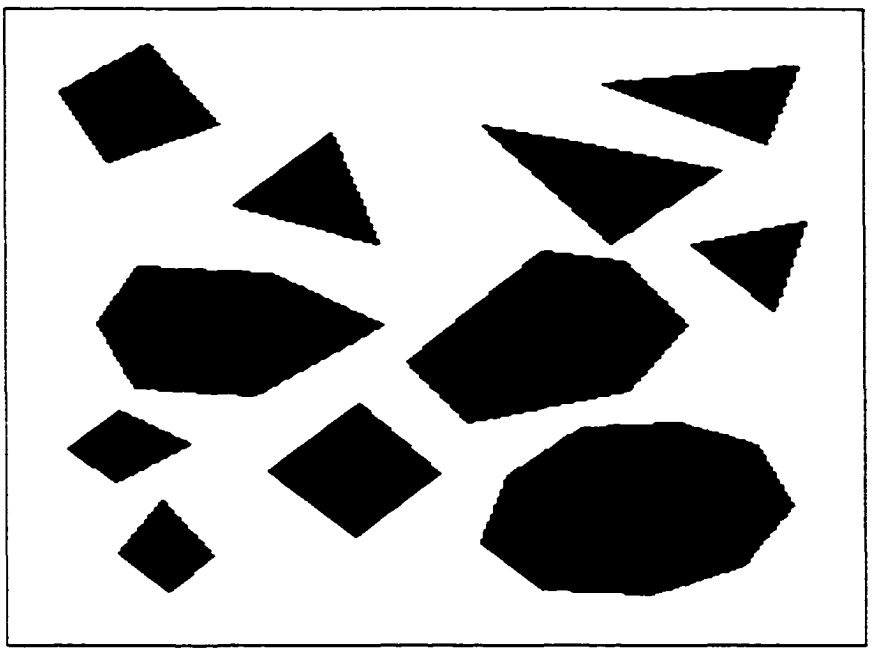


Figure 104 : Scène de référence illustrant le processus de planification de trajectoire

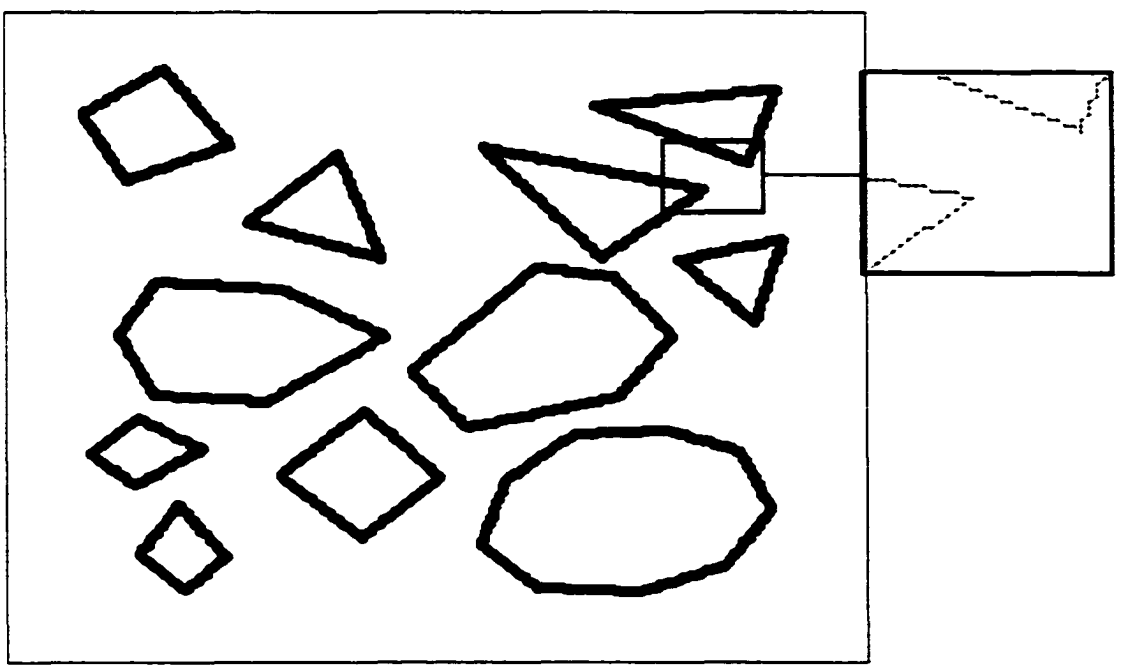


Figure 105 : Polygonisation de la scène illustrée à la figure 104

9.3 Réduction du nombre de sommets des polygones

Dans le but de minimiser la complexité des polygones, on réduit le nombre de sommets décrivant le contour de ceux-ci. Cette compression des données est essentielle pour réduire le nombre de chemins possibles. La tâche consiste alors à éliminer tous les éléments milieux des segments linéaires. Par contre, on laisse inchangés les éléments du contour décrivant des formes différentes des droites. Il existe un nombre considérable de façons de réaliser cette tâche. On présente ici deux méthodes : la première est basée sur une suite de droites moyennes au point d'analyse et la deuxième est basée sur l'analyse de l'angle que font les vecteurs des n points suivants et précédents. Il ne faut pas perdre de vue que ces méthodes ne font qu'approximer le contour réel des polygones par un contour réduit.

9.3.1 Méthode des droites moyennes

On pose S_j le $j^{ième}$ sommet du contour du polygone P. L'idée consiste à prendre les k points contigus à S_j (de S_j à S_{j-k}) et de calculer $\bar{d}$ la distance moyenne de chaque sommet par rapport à la droite minimisant la distance à chaque point. On commence avec $k = 2$ et si $\bar{d}$ est inférieur à la limite de tolérance d_{min}, on recommence en incrémentant k. Dès que $\bar{d}$ est supérieur à d_{min}, on supprime tous les sommets inclus dans l'intervalle $[j+1, j+k-1]$.

La droite minimisant la distance à chaque point peut se calculer par une régression linaire dont l'équation est la suivante :

$$\hat{y}_i = \overline{m} \cdot x_i + \overline{b} \tag{9.3}$$

où : $\hat{y}$ est l'estimation de y faite par la régression linéaire

$\overline{m}$ est la pente de la droite minimisant la distance à chaque point

$$\overline{m} = \frac{n \cdot \sum_{i=j}^{j+k} x_i y_i - \sum_{i=j}^{j+k} x_i \cdot \sum_{i=j}^{j+k} y_i}{n \cdot \sum_{i=j}^{j+k} x_i^2 - \left(\sum_{i=j}^{j+k} x_i \right)^2}$$

$\overline{b}$ est l'ordonnée à l'origine de la droite minimisant la distance à chaque point

$$\overline{b} = \frac{\sum_{i=j}^{j+k} y_i - \overline{m} \cdot \sum_{i=j}^{j+k} x_i}{n}$$

n est le nombre de points

j est le point de référence

k est le nombre de points contigus sélectionnés pour la droite de régression linéaire

(x_i, y_i) est la coordonnée du $i^{ième}$ sommet

L'indice d'erreur se calcule par la distance moyenne des sommets (S_j à S_{j+k}) par rapport à la droite de régression linéaire.

$$\overline{d} = \frac{\sum_{i=j}^{j+k} \frac{\| v_d \wedge v_{s_i} \|}{\| v_d \|}}{k+1} \tag{9.4}$$

où : $\|a\|$ est la norme du vecteur a. $\|a\| = \sqrt{a_x^2 + a_y^2}$

v_d est le vecteur directeur de la droite de régression linéaire. $v_d = \begin{bmatrix} \frac{-\overline{b}}{\overline{m}} & -\overline{b} \end{bmatrix}$

v_{s_i} est le vecteur directeur du $i^{ième}$ sommet par rapport à v_d. $v_{s_i} = \begin{bmatrix} s_{i_x} & s_{i_y} - \overline{b} \end{bmatrix}$

$\wedge$ représente le produit vectoriel

Cette méthode simple donne de bons résultats mais réduit la taille réelle du polygone en tronquant les coins. De plus, la méthode utilisée ici est basée sur la description d'une droite selon les coordonnées rectangulaires. Or dès que les sommets analysés s'alignent verticalement, $\overline{m} \to \infty$, ce qui engendre des erreurs et des cas de gestion d'exception

pour l'algorithme. Le tableau suivant est un exemple illustrant cet algorithme étape par étape (pour les étapes où k = 2, 5, 8 et 11). Les points —•— indiquent un sommet du polygone et les points —•— indiquent un sommet sélectionné pour le calcul de la régression linéaire et de $\bar{d}$.

Tableau XII

Exemple de réduction du nombre de sommets d'un polygone

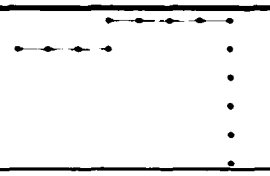
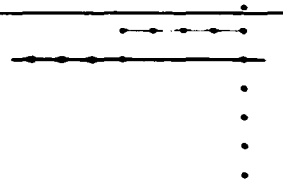
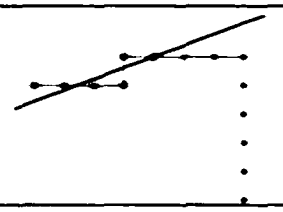
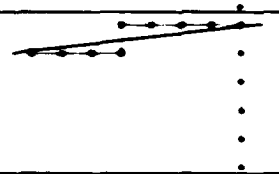
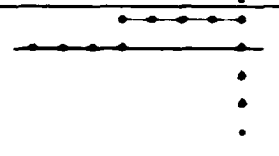
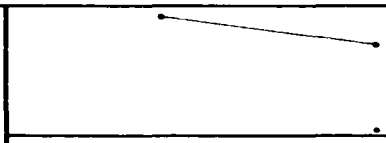
Étape	Segment de polygone	Description
0		Exemple de contour
1		$j = 1$ $k = 2$ $\bar{d} < d_{max}$
2		$j = 1$ $k = 5$ $\bar{d} < d_{max}$
3		$j = 1$ $k = 8$ $\bar{d} < d_{max}$
4		$j = 1$ $k = 11$ $\bar{d} > d_{max}$

Tableau XII (suite)

5		Résultat obtenu - coin mal évalué -
		Résultat souhaité

Cet algorithme appliqué à la figure 105 donne :

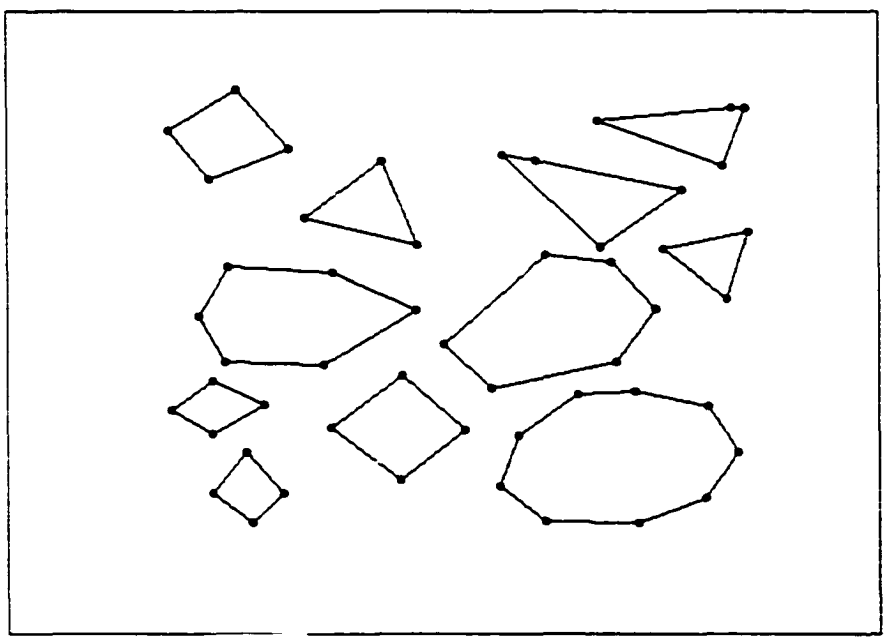


Figure 106 : Réduction du nombre de sommets des polygones illustrés à la figure 105

9.3.2 Méthode de l'angle avec les vecteurs précédents et suivants

On pose S_j le $j^{ème}$ sommet du contour du polygone P. L'idée consiste à calculer le vecteur directeur V_p formé par les k points contigus précédents S_j (de S_j à S_{j-k}) ainsi que le vecteur V_s formé par les k points contigus suivants S_j (de S_j à S_{j+k}) et de calculer

l'angle que font les vecteurs V_p et V_s. Si l'angle est compris dans un intervalle défini, alors le sommet est retiré de la liste des sommets décrivant le polygone. Cette technique a l'avantage de ne pas tronquer les coins mais elle possède aussi le désavantage de réduire les zones légèrement curvilignes selon les intervalles définis. Malgré tout, on peut l'utiliser pour réduire avantageusement les zones concaves, ce qui est souvent souhaité lors de la réduction d'un polygone.

On calcule V_p et V_s en faisant la moyenne des composantes x et y des coordonnées des k sommets considérés.

$$V_p = \left(\frac{\bar{x}_p}{\sqrt{\bar{x}_p^{\,2} + \bar{y}_p^{\,2}}}, \frac{\bar{y}_p}{\sqrt{\bar{x}_p^{\,2} + \bar{y}_p^{\,2}}} \right)$$
$$V_s = \left(\frac{\bar{x}_s}{\sqrt{\bar{x}_s^{\,2} + \bar{y}_s^{\,2}}}, \frac{\bar{y}_s}{\sqrt{\bar{x}_s^{\,2} + \bar{y}_s^{\,2}}} \right) \qquad (9.5)$$

où : $\bar{x}_p = \dfrac{\sum_{m=0}^{k} S^x_{i-m}}{k+1}$ et $\bar{y}_p = \dfrac{\sum_{m=0}^{k} S^y_{i-m}}{k+1}$

$$\bar{x}_s = \frac{\sum_{m=0}^{k} S^x_{i+m}}{k+1} \text{ et } \bar{y}_s = \frac{\sum_{m=0}^{k} S^y_{i+m}}{k+1}$$

On calcule l'angle formé par les deux vecteurs directeurs θ ainsi :

$$\theta = \arccos\left(V_p \wedge V_s\right) \qquad (9.6)$$

9.4 Création des polygones convexes

L'utilisation de polygones convexes permet de simplifier considérablement les problèmes associés à la manipulation de polygones. On définit un polygone convexe

ainsi : il est impossible de tracer une droite à l'intérieur du polygone sans traverser son contour. On remarque que tous les angles intérieurs du polygone sont inférieurs à 180°.

a) Polygone convexe b) Polygone concave

Figure 107 : Exemple de polygones convexe et concave

Les données brutes de la RVE donnent généralement plusieurs polygones concaves. Il est important de trouver une technique permettant de scinder proprement ces polygones afin d'obtenir plusieurs polygones convexes. Il existe plusieurs algorithmes permettant de créer des polygones convexes. Celui proposé ici est une approche récursive du problème. Lorsqu'un polygone est concave, on le simplifie par la création d'un polygone convexe et d'un polygone de forme indéfinie. On applique récursivement cet algorithme sur les polygones indéfinis tant qu'ils sont concaves. Voici les étapes principales de l'algorithme.

1. On identifie tous les sommets S_i correspondant aux concavités C_i du polygone P. Pour ce faire, on utilise la propriété des vecteurs traversables. On calcule Q en utilisant le segment $\overline{S_{i-1}S_i}$ au lieu d'un polygone et le point S_{i-1} comme point d'intérêt. Si Q est égal à 1, on a un sommet convexe. Dans le cas contraire, on obtient un somment concave. Il est important que S_{i-1}, S_i et S_{i+1} soient définis dans le sens anti-horaire.

2. On évalue si le polygone est concave par le nombre de sommets correspondant aux concavités n_C. Si $n_C = 0$, le polygone est convexe et on arrête le processus

sinon on divise le polygone concave en deux polygones : un polygone convexe et un polygone indéfini. La division se fait par les étapes suivantes.

3. On identifie un sommet S_i pour lequel il existe deux voisins consécutifs, S_{i-d} et S_{i-2d} (où d est la direction du voisin considéré, c'est-à-dire 1 ou -1), qui respectent les trois critères suivants :
   i. le sommet S_{i-d} est un sommet convexe (ceci implique que le polygone formé par les sommets S_i - S_{i-d} - S_{i-2d} est un polygone convexe);
   ii. le sommet S_{i-3d} est à l'extérieur (la frontière étant incluse) du polygone formé par les sommets S_i - S_{i-d} - S_{i-2d} (on utilise la technique des vecteurs traversables pour réaliser cette partie);
   iii. le segment formé par les sommets S_{i-2d} et S_i n'entre pas en collision avec le polygone global P (on utilise encore la technique des vecteurs traversables pour réaliser cette partie).
4. On parcourt les sommets du polygone à partir de S_{i-3d} jusqu'à ce qu'on atteigne un sommet $S_{i-j\cdot d}$ qui correspond à l'un des quatre critères suivants :
   i. $S_{i-j\cdot d}$ est un sommet correspondant à une concavité;
   ii. le sommet $S_{i-(j-1)\cdot d}$ est à l'intérieur (la frontière étant exclue) du polygone formé par les sommets S_i à $S_{i-j\cdot d}$ (on utilise la technique des vecteurs traversables pour réaliser cette partie);
   iii. le segment formé par les sommets $S_{i-(j-1)\cdot d}$ et S_i entre en collision avec le polygone global (les vecteurs traversables sont utilisés pour résoudre ce problème);
   iv. le polygone créé par les sommets S_i à $S_{i-(j-1)\cdot d}$ est concave (on utilise encore la technique des vecteurs traversables pour réaliser cette partie).
5. On scinde le polygone concave P en deux polygones : le polygone convexe $P1$ formé par les sommets S_i à $S_{i-j\cdot d}$ et le polygone de forme indéfinie $P2$ formé par

les sommets $S_{i-(j-1)\cdot d}$ à S_i (en respectant la définition circulaire des sommets décrivant les polygones).

6. On recommence récursivement le processus avec *P2*.

La nature récursive de cet algorithme permet d'utiliser une structure de données simple et performante. Cette structure est constituée d'une classe (dans le cas d'une programmation par objet) définissant un polygone. On ajoute à cette classe deux pointeurs permettant d'identifier deux sous-polygones : un polygone de forme convexe et l'autre de forme indéfinie. Ainsi, lorsqu'un polygone est concave, il se scinde en deux parties explicitement définies par des sous-polygones. De cette façon, l'algorithme qui scinde le polygone est appelé récursivement pour chaque polygone indéfini tant que les sous-polygones indéfinis sont concaves.

Le tableau de la page suivante présente étape par étape cet algorithme.

Tableau XIII

Exemple de création de polygones convexes

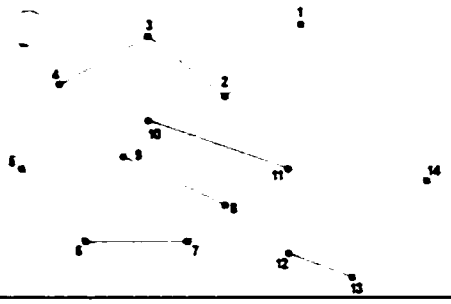
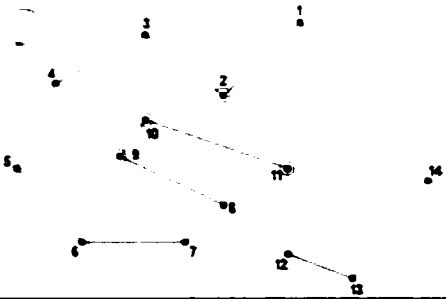
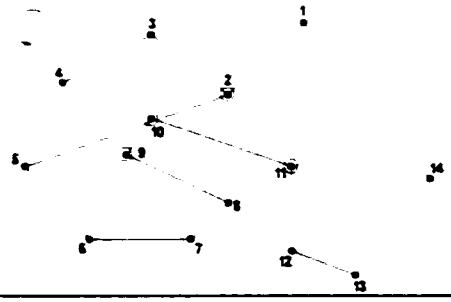
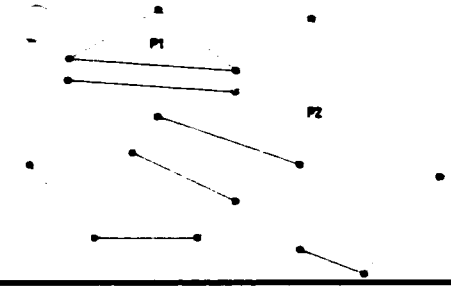
Appel récursif	Étape	Segment de polygone	Description
1	0		Polygone à traiter.
	1–2–3		1. Évaluation de tous les sommets correspondants aux concavités. 2. $n_C = 4 > 0 \Rightarrow$ Polygone concave 3. $i = 2$ et $d = +1$
	4		On parcourt le polygone à la recherche d'une contrainte. Au sommet 5, c'est la contrainte 3 qui apparaît.
	5		Division du polygone *P* en deux polygones (*P1* et *P2*).

Tableau XIII (suite)

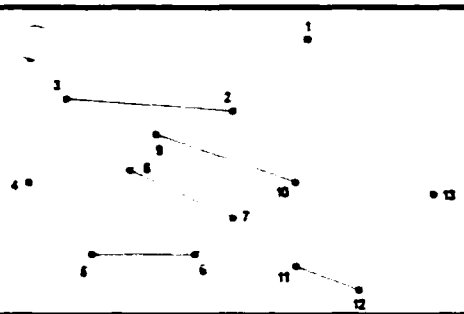
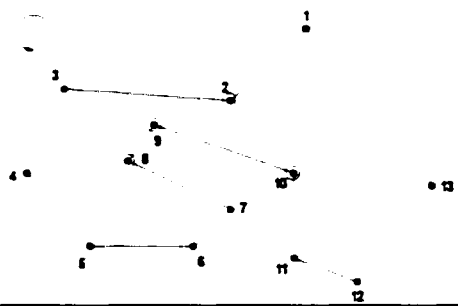
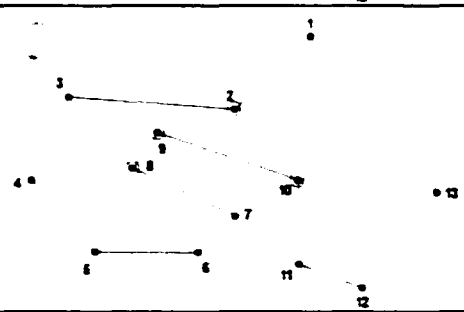
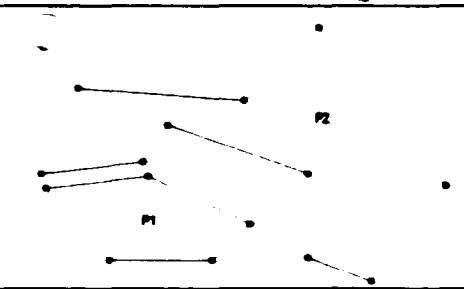
Appel récursif	Étape	Segment de polygone	Description
2	0		Polygone à traiter.
	1–2–3		1. Évaluation de tous les sommets correspondants aux concavités. 2. $n_C = 4 > 0 \Rightarrow$ Polygone concave 3. $i = 8$ et $d = -1$
	4		On parcourt le polygone à la recherche d'une contrainte. Au sommet 3, c'est la contrainte 4 qui apparaît.
	5		Division du polygone *P* en deux polygones (*P1* et *P2*).

Tableau XIII (suite)

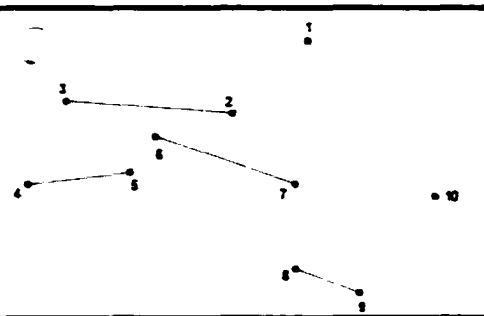
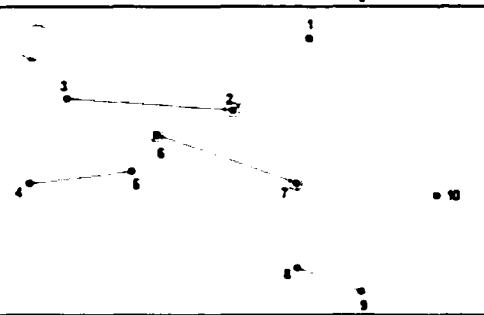
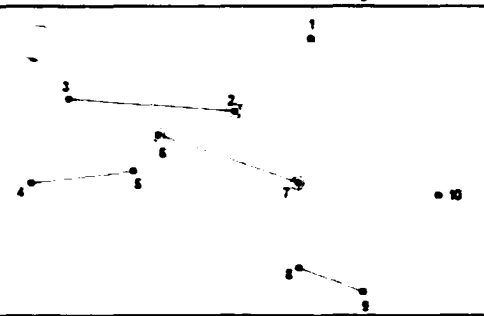
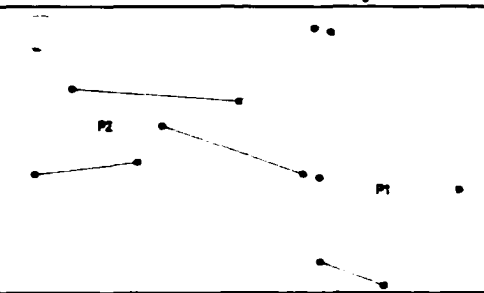
Appel récursif	Étape	Segment de polygone	Description
3	0		Polygone à traiter.
	1–2–3		1. Évaluation de tous les sommets correspondants aux concavités. 2. $n_C = 3 > 0 \Rightarrow$ Polygone concave 3. $i = 7$ et $d = +1$
	4		On parcourt le polygone à la recherche d'une contrainte. Au sommet 2, c'est la contrainte 4 qui apparaît.
	5		Division du polygone *P* en deux polygones (*P1* et *P2*).

Tableau XIII (suite)

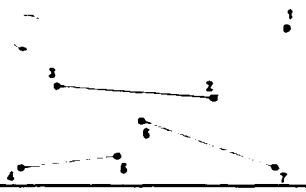
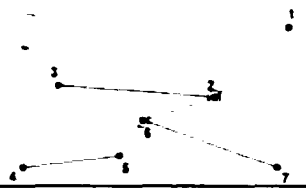
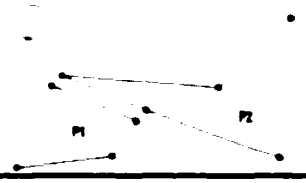
Appel récursif	Étape	Segment de polygone	Description
4	0		Polygone à traiter.
	1–2–3		1. Évaluation de tous les sommets correspondants aux concavités. 2. $n_C = 2 > 0 \Rightarrow$ Polygone concave 3. $i = 6$ et $d = -1$
	4		On parcourt le polygone à la recherche d'une contrainte. Au sommet 2, ce sont les contraintes 1 et 4 qui apparaissent.
	5		Division du polygone *P* en deux polygones (*P1* et *P2*).

Tableau XIII (suite)

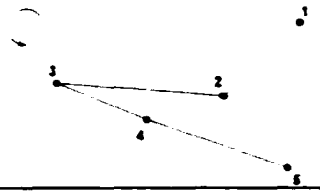
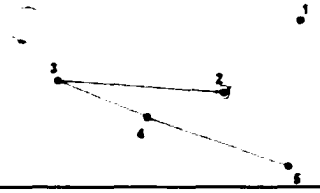
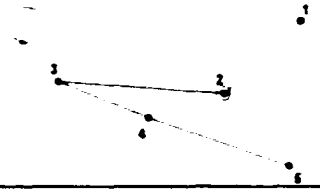
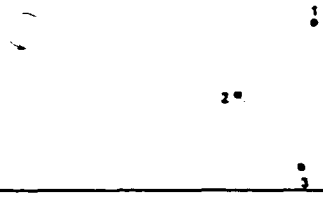
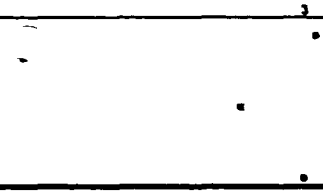
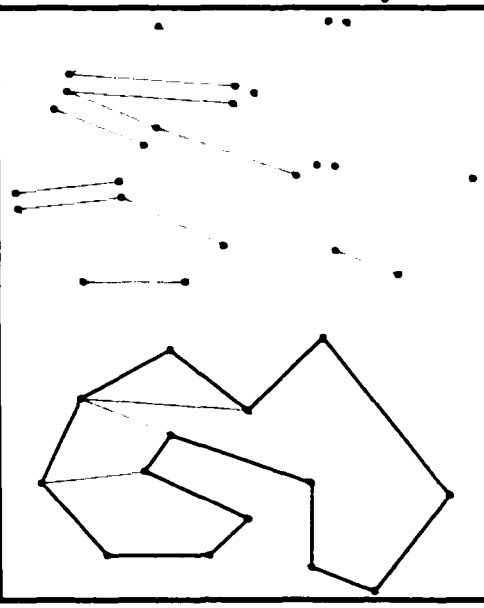
5	0		Polygone à traiter.
	1–2–3		1. Évaluation de tous les sommets correspondants aux concavités. 2. $n_C = 1 > 0 \Rightarrow$ Polygone concave 3. $i = 2$ et $d = +1$
	4		On parcourt le polygone à la recherche d'une contrainte. Au sommet 1, c'est la contrainte 4 qui apparaît.
	5		Division du polygone *P* en deux polygones (*P1* et *P2*).

Tableau XIII (suite)

Appel récursif	Étape	Segment de polygone	Description
6	0		Polygone à traiter.
	1–2		1. Évaluation de tous les sommets correspondants aux concavités. 2. $n_C = 0 \Rightarrow$ Polygone convexe **L'algorithme termine**
Résultat			On obtient 6 polygones convexes qui, mis côte à côte, représentent le polygone initial.

La figure suivante illustre la structure de données obtenue selon l'implantation d'une classe récursive.

Figure 108 : Exemple de structure de données récursive obtenue

9.5 Croissance des polygones

Pour que le robot puisse être considéré comme un point dans l'espace libre, il faut augmenter la taille des obstacles en fonction de la dimension du robot. Dans la littérature, on utilise systématiquement une méthode consistant à construire le nouveau polygone à partir de la liste des faces, des sommets et des segments. On présente ici une méthode plus simple et plus rapide à calculer. Cette technique consiste à placer un polygone identique à celui décrivant le robot selon chacun de ses sommets sur tous les sommets du polygone décrivant l'obstacle. On élimine ensuite les polygones correspondant au robot qui sont en collision avec le polygone correspondant à l'obstacle (on utilise la méthode des vecteurs traversables pour déterminer si un polygone entre en collision avec un autre). Finalement, le polygone modifié décrivant l'obstacle est défini en liant les sommets de référence des polygones correspondant au robot.

Le tableau suivant illustre cet algorithme étape par étape.

Tableau XIV

Exemple de croissance d'un obstacle

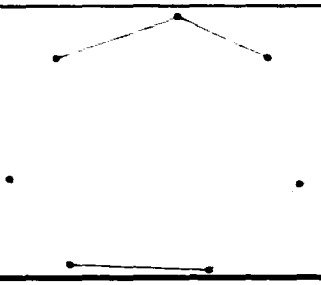
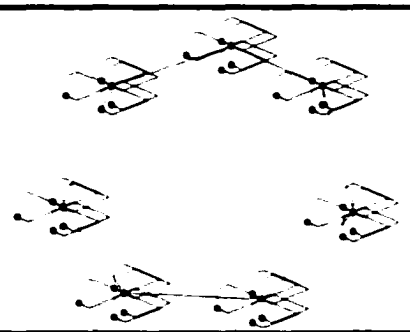
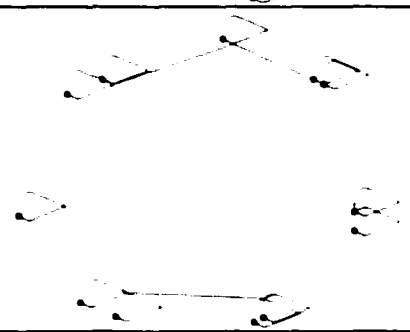
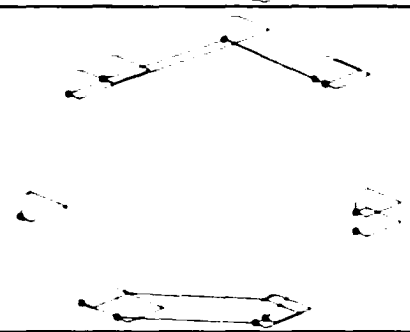
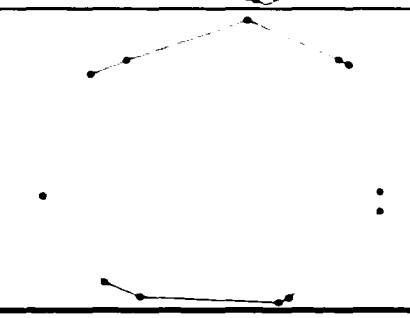
Étape	Graphique	Description
0		Description polygonale du robot
		Description polygonale de l'obstacle

Tableau XIV (suite)

1		Positionnement des polygones représentant le robot. *Cette étape consiste à positionner des polygones décrivant le robot selon tous leurs sommets sur tous les sommets du polygone représentant l'obstacle.*
2		Élimination de quelques polygones représentant le robot *Cette étape consiste à retirer tous les polygones représentant le robot qui entrent en collision avec le polygone représentant l'obstacle.*
3		Construction du polygone final. *Cette étape consiste à lier tous les points de références des polygones correspondant au robot.*
4		Polygone final

La figure suivante montre le résultat de cet algorithme sur la description polygonale de la scène étudiée.

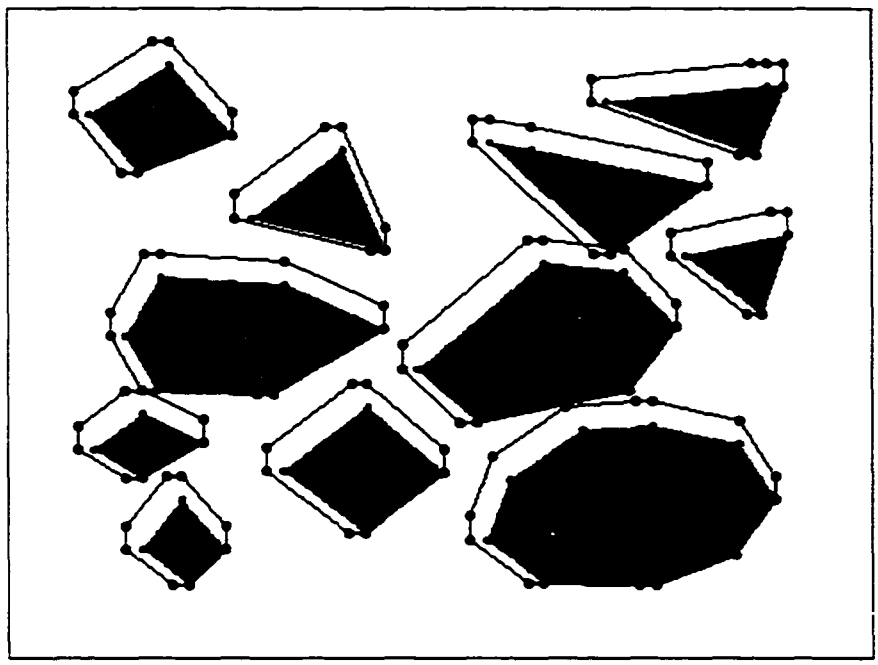


Figure 109 : Exemple de croissance des obstacles pour les polygones de la scène illustrée à la figure 106

9.6 Identification des chemins possibles et réduction par les chemins tangents

Le chemin optimal passe inévitablement par quelques sommets des différents polygones présents dans l'environnement. Les segments de chemin c_i sont définis par les segments pouvant être créés à partir de tous les sommets des n polygones. Puisque les segments représentent les chemins, on limite ces segments à ceux qui n'entrent pas en collision avec les différents polygones. On utilise encore une fois les vecteurs traversables pour déterminer si un segment entre en collision avec les polygones de la scène. En plus de considérer tous les sommets de tous les polygones, on doit aussi considérer la position initiale du robot et la position de destination du robot. Sur la figure suivante, ces positions sont illustrées en haut à gauche pour le point de départ et en bas à droite pour le point d'arrivée.

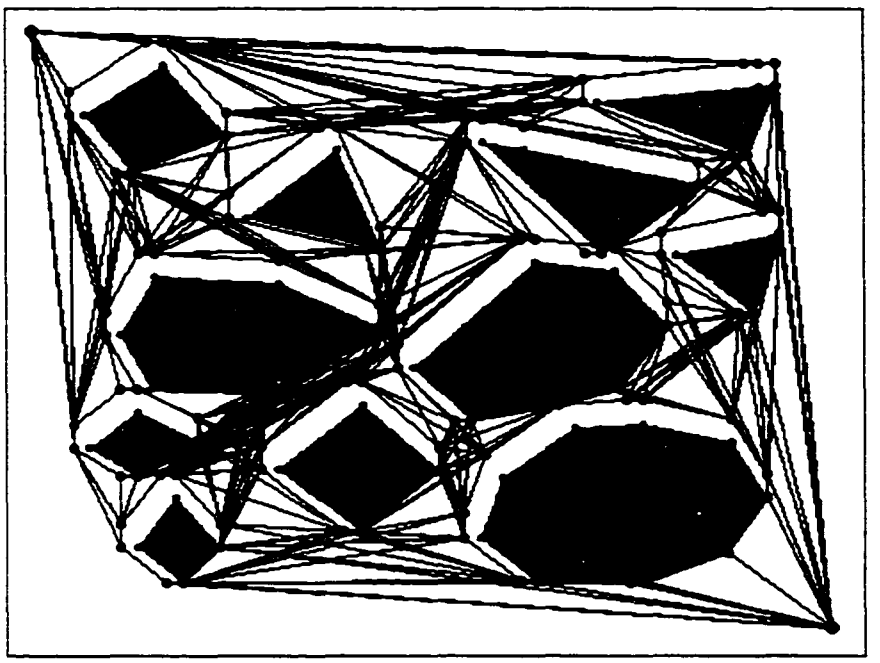


Figure 110 : Exemple d'identification de tous les chemins possibles pour la scène illustrée à la figure 107

On remarque que le nombre de segments de chemin est considérable et que plusieurs de ces segments sont inutiles. En effet, les seuls segments à considérer sont les segments tangents aux obstacles. La figure suivante montre deux exemples simples du concept des segments de chemin tangents aux obstacles.

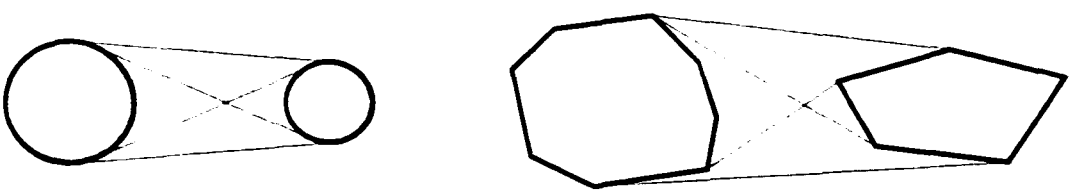


Figure 111 : Exemples de chemins tangents entre deux obstacles

Encore une fois, on peut utiliser certaines propriétés des vecteurs traversables pour aider à identifier ces segments.

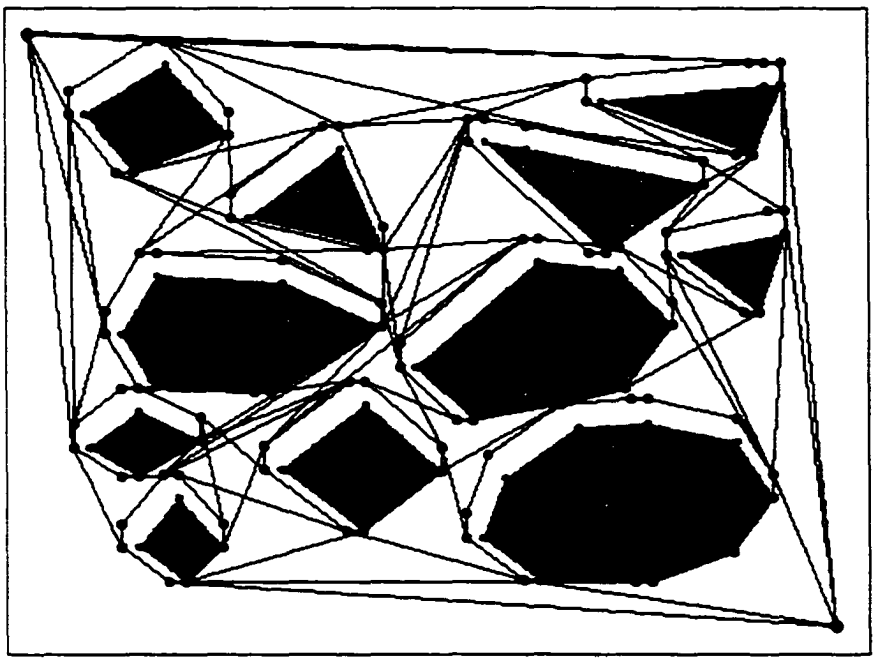


Figure 112 : Réduction du nombre de chemins possibles par les chemins tangents

L'étape de réduction des chemins n'est pas essentielle pour solutionner le problème mais réduit considérablement le temps de calcul requis pour déterminer le chemin le plus court.

9.7 Calcul des coûts associés à chaque segment de chemin

Avant de déterminer quel chemin est le plus court, il est important de définir la longueur de chacun des segments de chemin. Il est possible de considérer d'autres critères en plus de la distance entre chaque sommet. Par exemple, on peut être intéressé de trouver le chemin qui passe le plus à droite possible et pondérer chacun des chemins en conséquence. Dans le cas d'un environnement 2D½, on mesure la distance entre chaque segment par le déplacement réel du robot sur la surface de la scène et non seulement la distance orthogonale au plan de représentation.

La méthode implantée utilise trois critères : la distance réelle sur la surface 2D½ et deux facteurs de pénalité (en fonction de la pente sur le terrain et des discontinuités

verticales). Ces derniers critères permettent de minimiser les risques d'erreur de parcours pour des chemins trop accidentés. Pour réaliser le calcul d'un tel coût, on doit faire une sommation sur tous les éléments du segment en considérant chaque critère utilisé. Le coût total se calcule par :

$$C_{\overline{S_i S_j}} = \alpha \cdot \tilde{d}_{\overline{S_i S_j}} + \beta \cdot \tilde{p}_{\overline{S_i S_j}} + \omega \cdot \tilde{t}_{\overline{S_i S_j}} \tag{9.7}$$

où : $C_{\overline{S_i S_j}}$ est le coût total du segment $\overline{S_i S_j}$

$\tilde{d}_{\overline{S_i S_j}}$ est la distance totale du segment $\overline{S_i S_j}$ sur la surface 2D½, $\tilde{d}_{\overline{S_i S_j}} = \sum_{k=0(\delta)}^{1} SE_{\bar{x}_\delta, \bar{y}_\delta}$

$\tilde{p}_{\overline{S_i S_j}}$ est une mesure de tous les éléments de pente normalisés le long du segment $\overline{S_i S_j}$ sur la surface 2D½. On propose deux méthodes de calcul qui considèrent tous les éléments de pente. La première méthode calcule la moyenne alors que la deuxième évalue la valeur maximum.

$$\tilde{p}_{\overline{S_i S_j}} = \delta \cdot \sum_{k=0(\delta)}^{1} \frac{CP_{\bar{x}_\delta, \bar{y}_\delta}}{\theta_{\max}} \quad \text{et} \quad \tilde{p}_{\overline{S_i S_j}} = \max\left[\frac{CP_{\bar{x}_\delta, \bar{y}_\delta}}{\theta_{\max}}\right]_{\forall \delta}$$

$\tilde{t}_{\overline{S_i S_j}}$ est une mesure de tous les éléments de discontinuité verticale normalisés le long du segment $\overline{S_i S_j}$ sur la surface 2D½. On propose deux méthodes de calcul qui considèrent tous les éléments de discontinuité verticale. La première méthode calcule la moyenne alors que la deuxième évalue la valeur maximum.

$$\tilde{t}_{\overline{S_i S_j}} = \delta \cdot \sum_{k=0(\delta)}^{1} \frac{CD_{\bar{x}_\delta, \bar{y}_\delta}}{\Delta y_{\max}} \quad \text{et} \quad \tilde{t}_{\overline{S_i S_j}} = \max\left[\frac{CD_{\bar{x}_\delta, \bar{y}_\delta}}{\Delta y_{\max}}\right]_{\forall \delta}$$

α, β et ω sont des facteurs de pondération pour les différents critères.

$SE_{\bar{x}_\delta, \bar{y}_\delta}$ est la surface d'élévation utilisée à la coordonnée $(\bar{\bar{x}}_\delta, \bar{\bar{y}}_\delta)$

$CP_{\bar{x}_\delta, \bar{y}_\delta}$ est la carte des pentes maximums non segmentée à la coordonnée $(\bar{\bar{x}}_\delta, \bar{\bar{y}}_\delta)$

$CD_{\bar{x}_\delta, \bar{y}_\delta}$ est la carte des discontinuités vert. non segmentée à la coordonnée $(\bar{\bar{x}}_\delta, \bar{\bar{y}}_\delta)$

θ_{max} est l'angle de la pente maximum toléré par le robot.

Δy_{max} est la hauteur maximum d'une discontinuité verticale tolérée par le robot.

$(\ddot{x}_\delta, \ddot{y}_\delta)$ est la coordonnée de la droite paramétrique formée par le segment $\overline{S_i S_j}$ à δ du sommet initial. $\ddot{x}_\delta = x_i + \delta \cdot (x_j - x_i)$ et $\ddot{y}_\delta = y_i + \delta \cdot (y_j - y_i)$

δ est le pas de l'équation paramétrique de la droite et indique le nombre d'éléments compris dans les différentes évaluations.

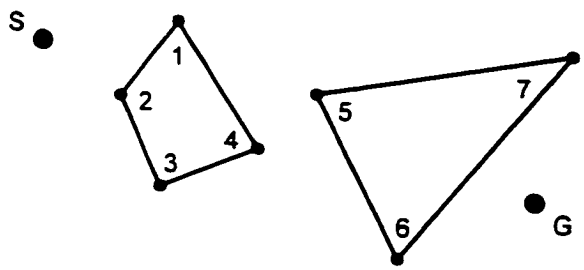
9.8 Sélection du chemin le plus court

L'algorithme de Dijkstra est une méthode par programmation dynamique permettant de simplifier le problème et de se rapprocher de la solution à chaque étape. Cet algorithme est certainement l'un des plus rapides pour trouver le chemin le plus court dans une structure de données semblable à celle obtenue.

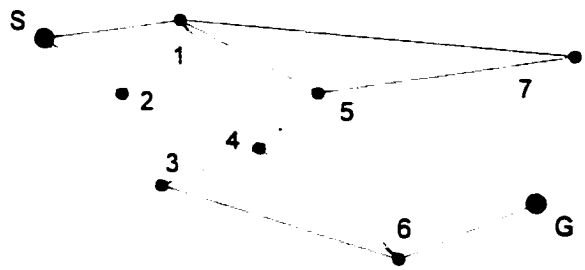
La méthode est simple et s'implante facilement. Elle passe par cinq étapes :

1. On crée la matrice de coûts correspondant à tous les chemins tangents.
2. On part du sommet correspondant au point de départ.
3. On calcule le coût incrémental de tous les sommets connectés au sommet courant : $C_{i-1} = C_i + C_{i-1}$. On nomme cette étape la relaxation.
4. On identifie le sommet le moins *coûteux*. On recommence à l'étape 3 tant que le sommet correspondant à la position d'arrivée n'est pas atteint.
5. On choisit le chemin optimal en prenant la ligne de l'élément sélectionné à rebours. On commence par G.

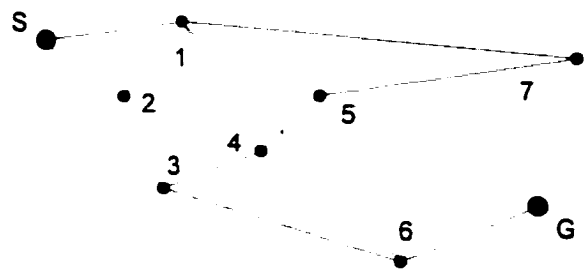
On propose une implantation sous forme de tableaux. Cette méthode permet une implantation particulièrement compacte de l'algorithme. On prend la scène suivante.



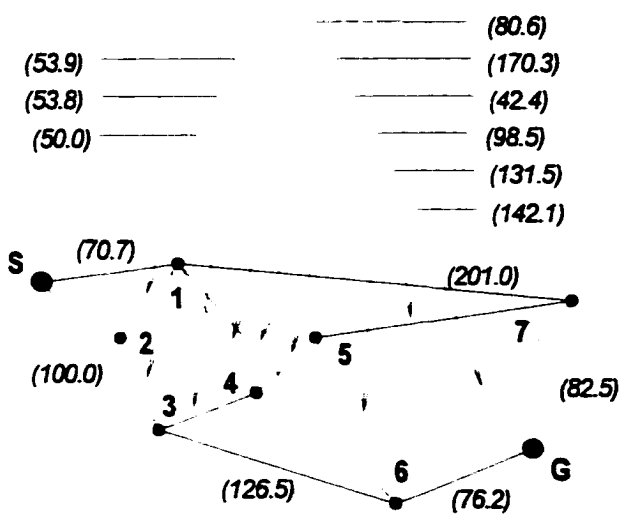
a) Scène simple (S et G représentent respectivement le point de départ et d'arrivée)



b) Tous les chemins possibles



c) Les chemins tangents



d) Coût de tous les chemins tangents

Figure 113 : Scène servant à illustrer l'implantation de l'algorithme de Dijkstra

Tableau XV

Exemple d'implantation de l'algorithme de Dijkstra

Étape 1

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	∞	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	∞	X	53,8	∞	∞	∞	∞	∞
3	∞	∞	∞	X	53,9	∞	126,5	∞	∞
4	∞	∞	∞	∞	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	∞	X	98,5	131,5	∞
6	∞	∞	∞	∞	∞	∞	X	142,1	76,2
7	∞	∞	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

Création de la matrice de coûts. Le triangle supérieur de la matrice est initialisé avec tous les coûts de chaque segment. Les chemins impossibles sont initialisés à l'infini. Le triangle inférieur est initialisé à l'infini.

2-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	∞	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	∞	X	53,9	∞	126,5	∞	∞
4	∞	∞	∞	∞	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	∞	X	98,5	131,5	∞
6	∞	∞	∞	∞	∞	∞	X	142,1	76,2
7	∞	∞	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

Le sommet d'intérêt est *S*.

On relaxe les éléments de la colonne sous *(S, S)*.

4

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	∞	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	∞	X	53,9	∞	126,5	∞	∞
4	∞	∞	∞	∞	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	∞	X	98,5	131,5	∞
6	∞	∞	∞	∞	∞	∞	X	142,1	76,2
7	∞	∞	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 1 correspond à ce choix.

Tableau XV (suite)

3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	*120,7*	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	∞	X	53,9	∞	126,5	∞	∞
4	∞	*151,3*	∞	∞	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	∞	X	98,5	131,5	∞
6	∞	*241,0*	∞	∞	∞	∞	X	142,1	76,2
7	∞	*271,7*	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

On relaxe les éléments de la colonne sous *(1, 1)*.

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	∞	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	*153,9*	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	∞	X	98,5	131,5	∞
6	∞	241,0	∞	*226,5*	∞	∞	X	142,1	76,2
7	∞	271,7	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 3 correspond à ce choix.

On relaxe les éléments de la colonne sous *(3, 3)*.

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	*174,5*	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	153,9	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	∞	X	98,5	131,5	∞
6	∞	241,0	∞	226,5	∞	∞	X	142,1	76,2
7	∞	271,7	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 2 correspond à ce choix.

On relaxe les éléments de la colonne sous *(2, 2)*.

Tableau XV (suite)

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	174,5	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	153,9	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	193,7	X	98,5	131,5	∞
6	∞	241,0	∞	226,5	∞	∞	X	142,1	76,2
7	∞	271,7	∞	∞	∞	∞	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 4 correspond à ce choix.

On relaxe les éléments de la colonne sous *(4, 4)*.

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	174,5	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	153,9	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	193,7	X	98,5	131,5	∞
6	∞	241,0	∞	226,5	∞	292,2	X	142,1	76,2
7	∞	271,7	∞	∞	∞	325,2	∞	X	82,5
G	∞	∞	∞	∞	∞	∞	∞	∞	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 5 correspond à ce choix.

On relaxe les éléments de la colonne sous *(5, 5)*.

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	174,5	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	153,9	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	193,7	X	98,5	131,5	∞
6	∞	241,0	∞	226,5	∞	292,2	X	142,1	76,2
7	∞	271,7	∞	∞	∞	325,2	368,6	X	82,5
G	∞	∞	∞	∞	∞	∞	302,7	∞	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 6 correspond à ce choix.

On relaxe les éléments de la colonne sous *(6, 6)*.

Tableau XV (suite)

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	174,5	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	153,9	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	193,7	X	98,5	131,5	∞
6	∞	241,0	∞	226,5	∞	282,2	X	142,1	76,2
7	∞	271,7	∞	∞	∞	325,2	368,6	X	82,5
G	∞	∞	∞	∞	∞	∞	302,7	354,2	X

On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet 7 correspond à ce choix.

On relaxe les éléments de la colonne sous *(7, 7)*.

4-3

	S	1	2	3	4	5	6	7	G
S	X	70,7	∞	100,0	∞	∞	∞	∞	∞
1	70,7	X	50,0	∞	80,6	∞	170,3	201,0	∞
2	∞	120,7	X	53,8	∞	∞	∞	∞	∞
3	100,0	∞	174,5	X	53,9	∞	126,5	∞	∞
4	∞	151,3	∞	153,9	X	42,4	∞	∞	∞
5	∞	∞	∞	∞	193,7	X	98,5	131,5	∞
6	∞	241,0	∞	226,5	∞	282,2	X	142,1	76,2
7	∞	271,7	∞	∞	∞	325,2	368,6	X	82,5
G	∞	∞	∞	∞	∞	∞	302,7	354,2	X

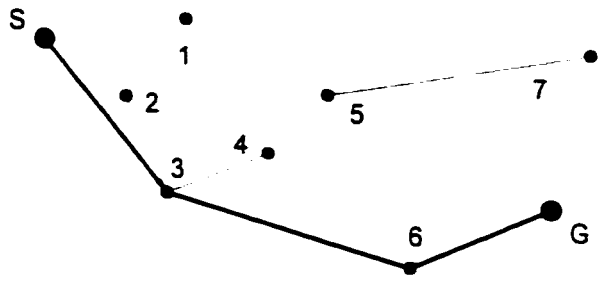
On choisit le chemin le moins coûteux dans le triangle inférieur. Le sommet G correspond à ce choix.

On relaxe les éléments de la colonne sous *(G, G)*.

5

S – 3 – 6 – G

(Point initial – 3ème sommet – 6ème sommet – Point d'arrivée)



On choisit le chemin optimal en prenant la colonne de l'élément sélectionné à rebours. On commence par G.

La figure suivante montre le chemin optimal parmi les chemins tangents de la figure 112.

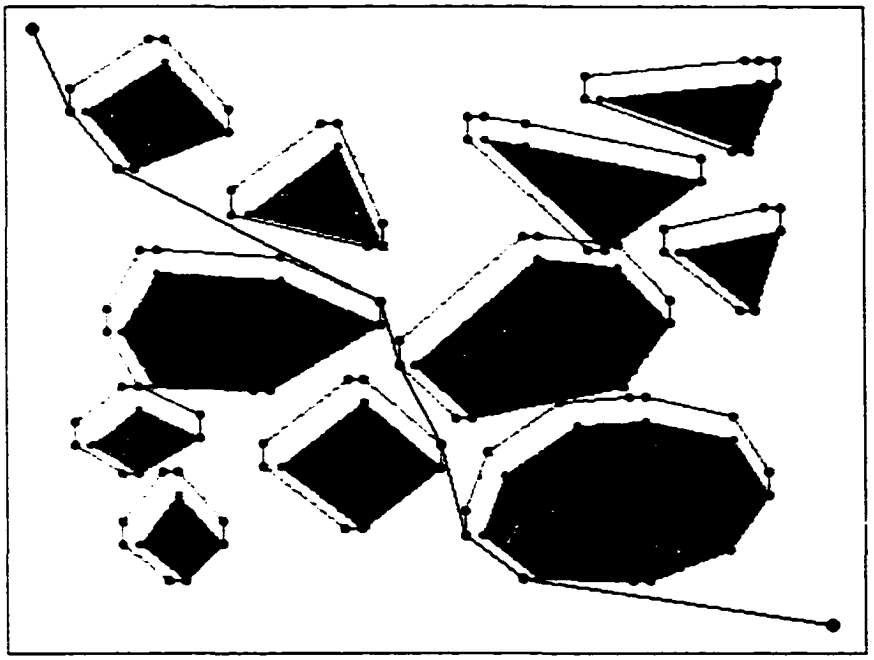


Figure 114 : Chemin optimal de la figure 112

CHAPITRE 10

SIMULATIONS

Puisque aucune plateforme mobile ne peut servir à valider simultanément tous les modules développés dans ce mémoire (perception de l'environnement, représentation virtuelle de l'environnement et planification de trajectoire), un simulateur est conçu. Ce simulateur permet le développement des algorithmes des modules de RVE et de planification de trajectoire en même temps que le développement de la mécanique et de la commande du robot. De plus, ce simulateur permet de valider les concepts développés ainsi que l'approche globale.

Le simulateur présente les caractéristiques suivantes :

- interface usager conviviale de haut niveau;
- affichage 3D de tous les éléments de la scène (avec la bibliothèque OpenGL comme moteur d'affichage 3D);
- affichage graphique de chacune des étapes du processus algorithmique;
- paramètres configurables;
- trajectoires définies selon l'usager ou en mode autonome.

Le simulateur fonctionne de la façon suivante :

1. une représentation synthétique d'un environnement réel est présentée au simulateur et sert de référence en tant que carte réelle;
2. le robot est positionné et se promène sur cette carte réelle;

3. l'intersection de tous les faisceaux laser est calculée avec la carte réelle selon les positions et orientations relatives : celles du capteur stéréoscopique par rapport au robot et celles du robot par rapport à la carte réelle;
4. chaque intersection entre un faisceau laser et la carte réelle correspond à un point à insérer dans le nuage de points;
5. pour chaque insertion de point dans le nuage de points, on met à jour les différentes représentations virtuelles intermédiaires pour obtenir la trajectoire finale;
6. finalement, le robot se meut selon la trajectoire définie.

Le simulateur suppose que le module de perception de l'environnement ne génère aucune erreur. Ainsi, on présume que le capteur stéréoscopique perçoit tous les points laser sans problème. De plus, on suppose connues et précises la position et l'orientation du robot par rapport à la carte virtuelle. Il est important de mentionner qu'à l'heure de la rédaction de ce mémoire, deux algorithmes ne sont pas implantés dans le simulateur, c'est-à-dire l'algorithme de croissance des polygones et l'algorithme de recherche des chemins tangents. Il faut garder en tête ces algorithmes manquants lors de l'analyse et de l'interprétation des résultats finals.

Ce chapitre présente les éléments de conception les plus importants du logiciel de simulation ainsi que la signification de chacune des étapes du processus. Enfin, plusieurs résultats sont illustrés.

10.1 Structure logicielle

Dans le but de structurer le logiciel et de lui assurer une grande flexibilité, les outils de conception logicielle sont mis à profit. Notamment, on utilise la méthode de conception UML à l'aide du logiciel Rational Rose™. Le simulateur est développé sous Windows™

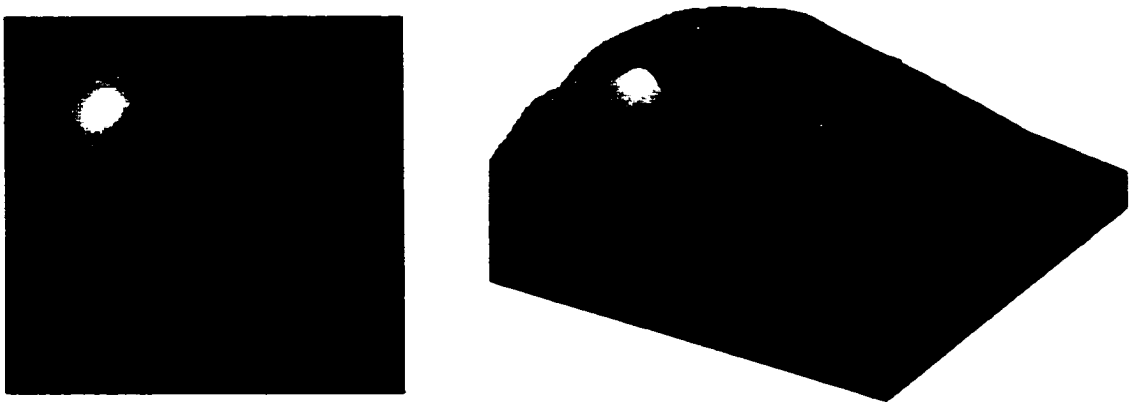
en langage C++ avec Borland C++Builder™. La bibliothèque graphique OpenGL™ est utilisée comme moteur d'affichage 3D.

Beaucoup de soins ont été apportés lors de la conception du logiciel pour rendre ce dernier robuste, efficace, modulaire et surtout très facile à modifier et améliorer. Ainsi, la structure logicielle actuelle rend facile l'implantation de nouvelles méthodes. De cette façon, il serait aisé d'ajouter de nouveaux outils ou de nouveaux algorithmes pouvant être mis à l'essai. Par exemple, il serait facile d'implanter de nouveaux algorithmes de planification de trajectoire et de comparer les performances de chacun d'entre eux selon différents critères.

10.2 Carte réelle ou environnement synthétique

La représentation synthétique utilisée est définie par une surface d'élévation à l'aide d'une image de format Bitmap. Ainsi, il est possible d'ouvrir une image de ce format où l'intensité du niveau de gris représente l'élévation de la carte réelle. L'environnement synthétique est de dimension égale à la taille de l'image ; une image de 680 × 480 pixels donne une surface de 680 × 480 unités.

On utilise ce format d'image normalisé afin de pouvoir créer facilement et rapidement des scènes complexes. Il est assez facile d'utiliser un logiciel de dessin 2D ou de retouches d'images comme Photoshop™ pour créer une carte synthétique correspondant à une scène à analyser



a) Image en niveau de gris (représentation 2D)

b) Représentation synthétique (représentation 3D)

Figure 115 : Image décrivant un environnement synthétique

10.3 Interface usager

L'interface usager est définie de façon telle qu'il soit très simple pour l'utilisateur de voir et de comprendre ce qui se passe à chaque étape du processus. Toutes les étapes importantes de calcul possèdent un mode de représentation spécifique. De cette façon, si un algorithme ou une méthode possède des failles, il est facile de les localiser. La figure 116 montre l'interface du simulateur et les points suivants décrivent ses principaux constituants :

1. menu et barres d'outils;
2. onglets permettant l'affichage de tous les paramètres configurables (*Carte réelle*, *Robot*, *Représentation virtuelle de l'environnement*, *Planification de trajectoire*, *Simulation* et *Divers*);
3. affichage de l'information (on voit ici la carte réelle);
4. affichage du robot, du laser, des faisceaux laser et des points laser faisant intersection avec la scène (le robot est représenté ici par un prisme rectangulaire bleu, les faisceaux laser par des lignes rosées et les points laser par des points rouges);

5. affichage du point d'arrivée représenté par un drapeau pour la simulation en mode autonome;
6. onglet permettant d'afficher les informations selon deux types de représentations: affichage graphique tel qu'illustré ici et affichage numérique des différents calculs;
7. barres d'information sur l'état instantané de plusieurs paramètres (position, orientation et facteur de zoom de l'affichage 3D, position et orientation du robot, nombre d'éléments pour les différentes représentations contenues dans la carte virtuelle, diverses informations sur les algorithmes de planification de trajectoire et informations générales reliées au contexte).

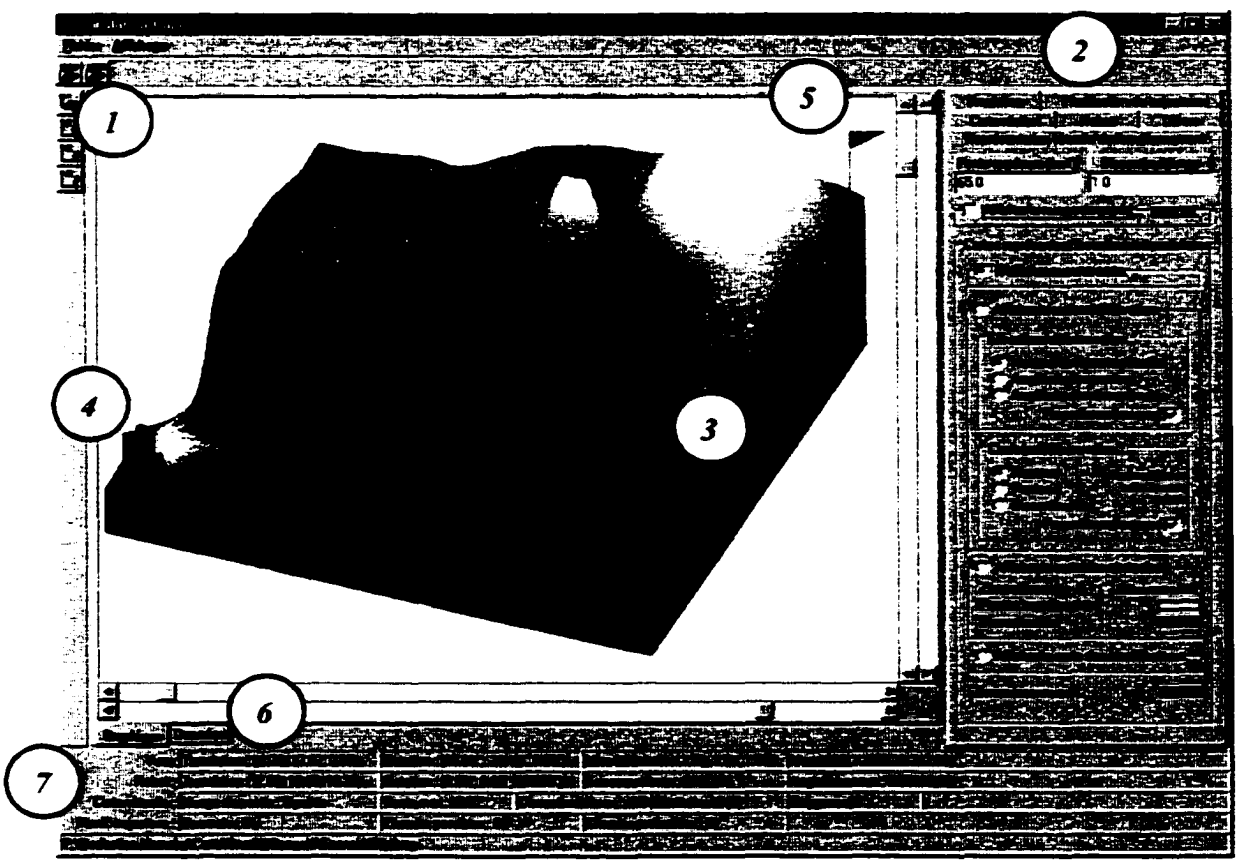
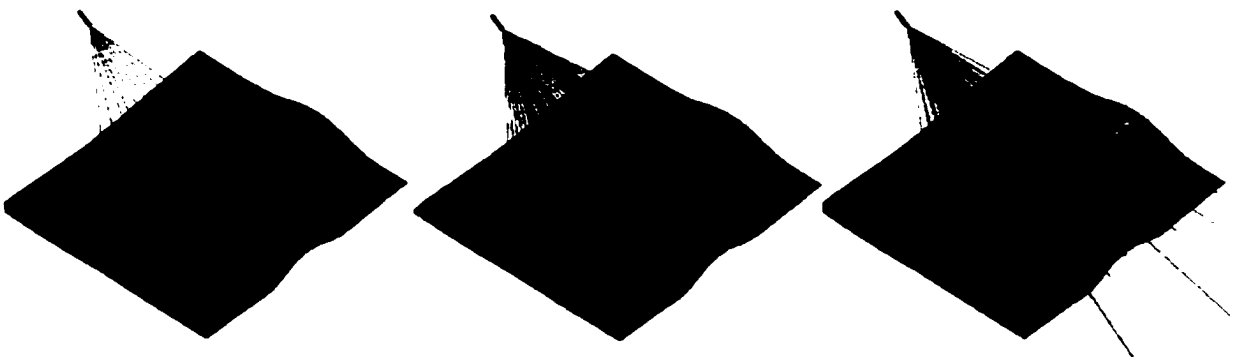


Figure 116 : Interface usager du simulateur

10.4 Configuration du robot et du capteur

Tous les paramètres importants décrivant le robot et le capteur stéréoscopique peuvent être configurés par l'usager. Celui-ci peut définir la dimension et la forme du robot selon un point de référence ainsi que la position et l'orientation du capteur stéréoscopique par rapport au robot. De plus, il est possible de définir le nombre de faisceaux laser dans les plans vertical et horizontal ainsi que la configuration de l'angle total d'ouverture des faisceaux laser pour les deux plans. Voici quelques exemples de configurations :



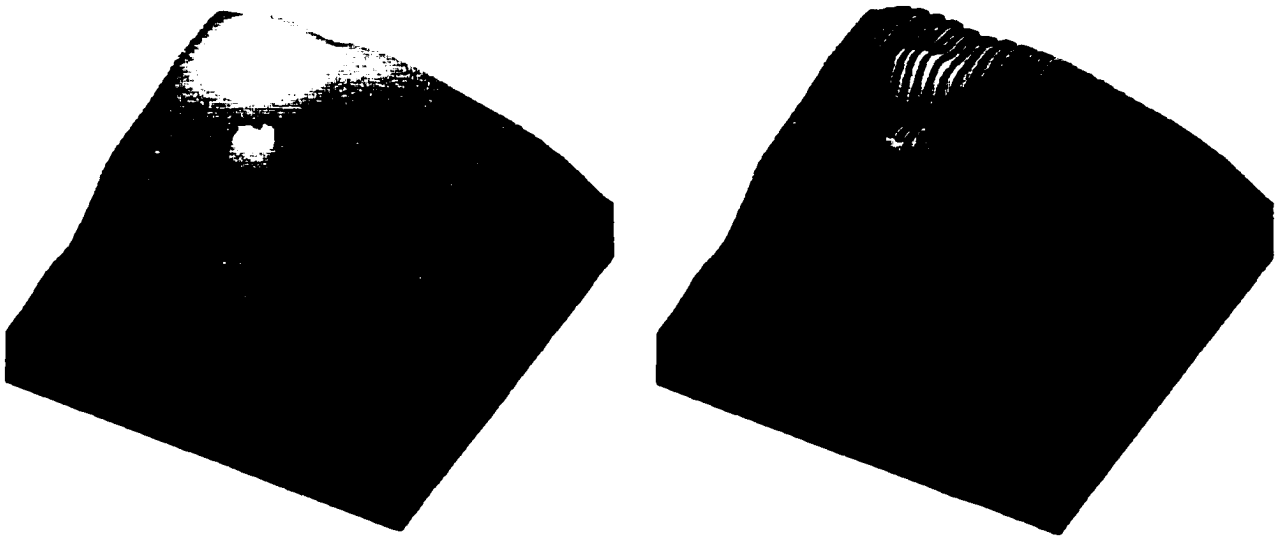
a) 3 × 3 faisceaux laser
20° × 30° d'ouverture totale

b) 7 x 7 faisceaux laser
20° × 30° d'ouverture totale

c) 7 × 7 faisceaux laser
30° × 20° d'ouverture totale

10.5 Description des étapes du processus algorithmique du module de RVE

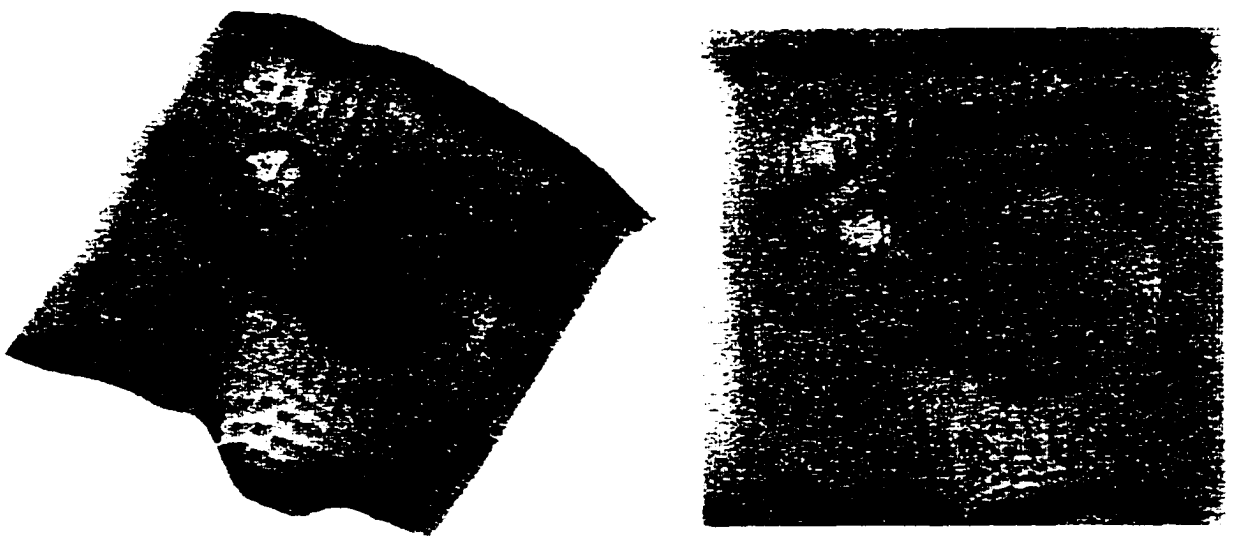
On montre ici toutes les étapes intermédiaires du processus algorithmique du module de RVE. Au début du processus, la carte réelle est inconnue du robot. Au fil de son déplacement, le robot accumule les points nécessaires à la construction du nuage de points. On calcule ensuite toutes les représentations intermédiaires jusqu'à la carte des obstacles. Les exemples suivants sont le résultat d'un déplacement linéaire du robot sur toute la surface du monde simulé. Le robot se déplace selon une trajectoire simple pour illustrer les méthodes de représentation intermédiaires.



a) Carte réelle b) Trajectoire empruntée par le robot

Figure 117 : Carte réelle et trajectoire empruntée par le robot

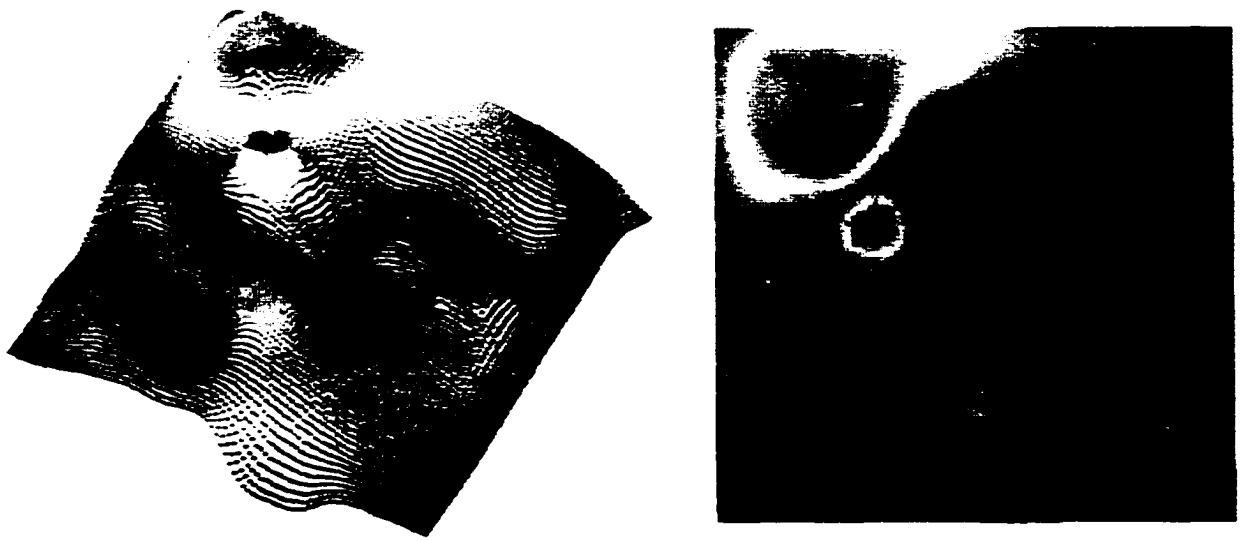
Pour chaque position du robot, on calcule l'intersection de tous les faisceaux laser avec la carte réelle. Ces points correspondent aux points d'insertion dans le nuage de points. La représentation en nuage de points suivante est très dense : on compte près de 370 000 points.



a) Vue isométrique b) Vue aérienne

Figure 118 : Nuage de points

Pour chaque insertion dans le nuage de points, on modifie la surface d'élévation en conséquence. On voit sur le graphique suivant la surface d'élévation illustrée par de petites surfaces carrées plates. Ces petites surfaces illustrent la discrétisation de l'environnement correspondant aux petites cellules de même dimension. La coloration utilisée suit une palette de couleur définie selon la teinte de l'espace de couleur HSV. Ainsi, le violet indique les régions les plus basses et le rouge indique les régions les plus élevées.



a) Vue isométrique

b) Vue aérienne

Figure 119 : Surface d'élévation

Pour chaque modification de la surface d'élévation, on évalue la carte des discontinuités verticales et la carte des pentes. Encore une fois, le violet représente une petite amplitude (petites discontinuités verticales et petites pentes) et le rouge représente une grande amplitude (grandes discontinuités verticales et pentes abruptes).

a) Vue isométrique b) Vue aérienne

Figure 120 : Carte des discontinuités verticales

a) Vue isométrique b) Vue aérienne

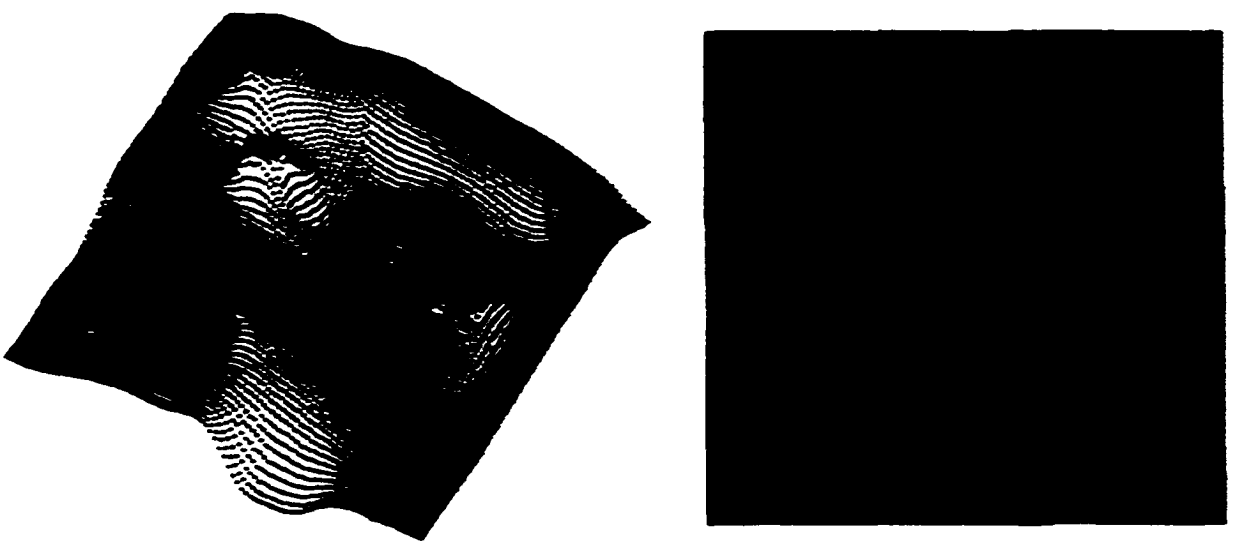
Figure 121 : Carte des pentes

Encore une fois, dès qu'une insertion est faite dans le nuage de points, on calcule la carte des obstacles et finalement la carte des identificateurs d'obstacles. Sur la carte des obstacles, le vert indique l'espace libre (où le robot peut circuler), le jaune les pentes

trop abruptes et le rouge les discontinuités verticales trop grandes. Évidemment, la définition de ces zones dépend des seuils de tolérance fixés pour les discontinuités verticales (Δy_{max}) et les pentes ($\Delta \theta_{max}$) maximums. La carte des identificateurs d'obstacles correspond à l'étape d'identification et de localisation des obstacles.

a) Vue isométrique b) Vue aérienne

Figure 122 : Carte des obstacles

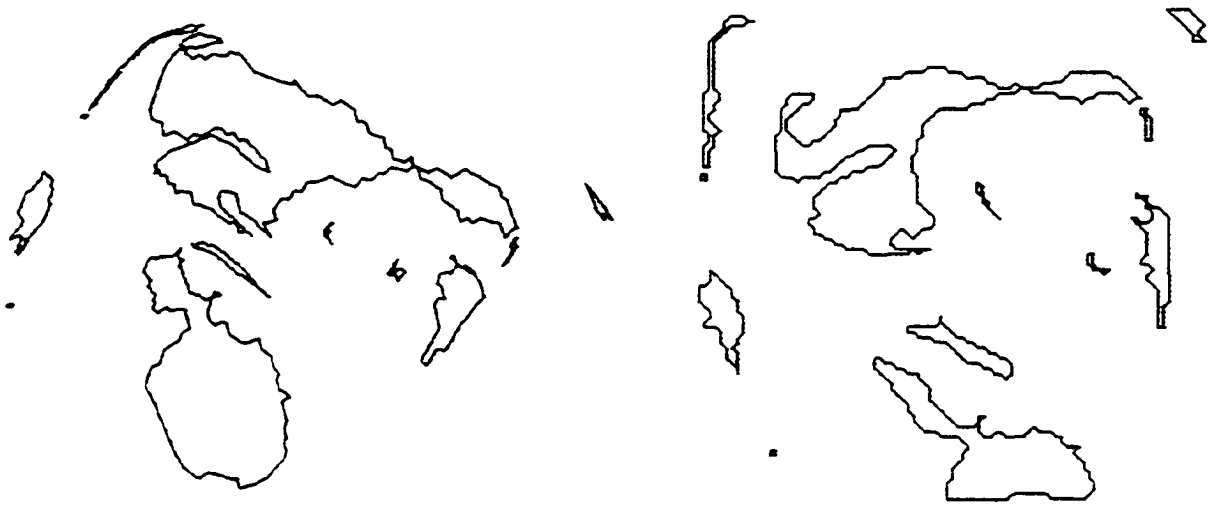


a) Vue isométrique b) Vue aérienne

Figure 123 : Carte des identificateurs d'obstacles

10.6 Description des étapes du processus algorithmique du module de planification de trajectoire

Lorsque la carte des identificateurs d'obstacles est évaluée, on passe à l'étape de la description des obstacles par des polygones. On remarque que les polygones sont définis en 3D. Les sommets sont indiqués en rouge et les arrêtes en bleu.



a) Vue isométrique b) Vue aérienne

Figure 124 : Polygones décrivant les obstacles

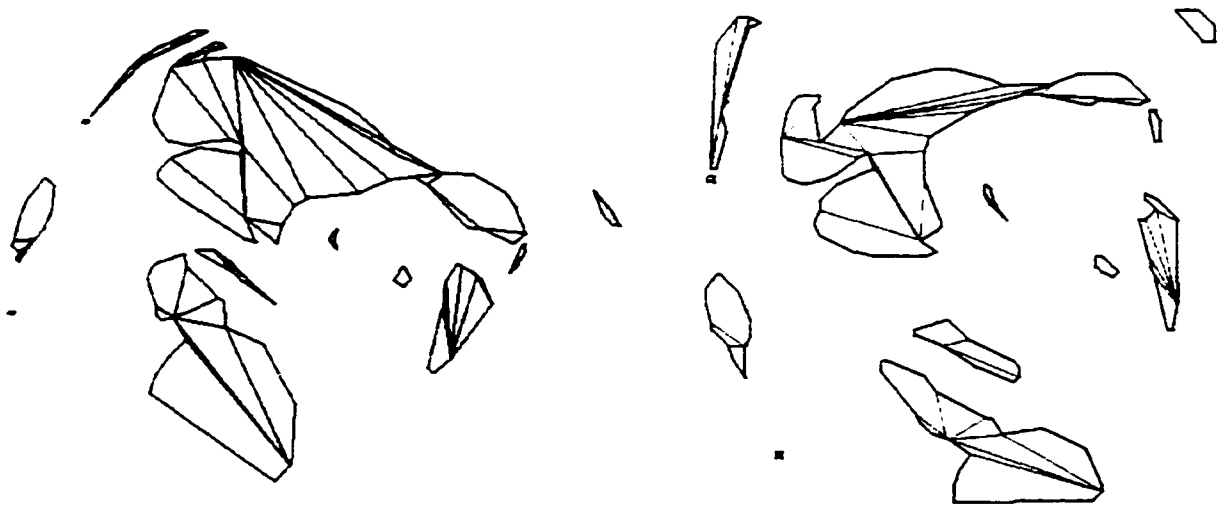
On peut passer maintenant à la réduction des polygones. La méthode utilisée ici correspond à celle présentée à la section 9.3.2. On applique cet algorithme trois fois successivement sur le contour avec les intervalles suivants qui permettent d'éliminer les sommets indésirables selon la valeur de l'angle θ (on utilise un $k = 2$) :

1. $\theta_1 \notin [220,360]$
2. $\theta_2 \in [170,190]$
3. $\theta_3 \in [175,185]$

a) Vue isométrique b) Vue aérienne

Figure 125 : Polygones réduits

Ayant maintenant une description polygonale avec un nombre réduit de points, on sépare les polygones concaves en polygones convexes.

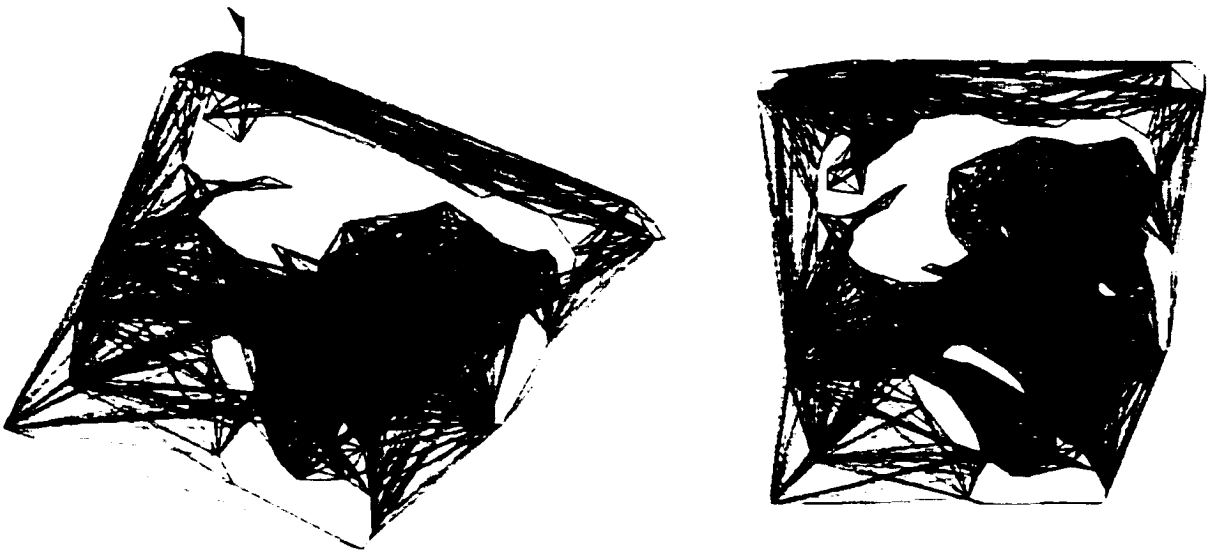


a) Vue isométrique b) Vue aérienne

Figure 126 : Polygones convexes

Comme il est mentionné, l'algorithme de croissance des polygones n'est pas implanté. Ainsi, la prochaine étape est l'identification des chemins praticables et le coût de ces

derniers. L'algorithme de recherche des chemins tangents n'étant pas implanté, on calcule le coût de tous les chemins praticables. Sur les figures suivantes, les chemins sont colorés en fonction de leur coût. La couleur violette indique un chemin ayant un faible coût et la couleur rouge indique un chemin dont le coût est élevé. On peut remarquer le drapeau indiquant l'objectif à atteindre. L'exemple de la prochaine figure montre des polygones décrits par 179 sommets créant ainsi 15 931 chemins possibles dont 2 558 sont praticables.



a) Vue isométrique b) Vue aérienne

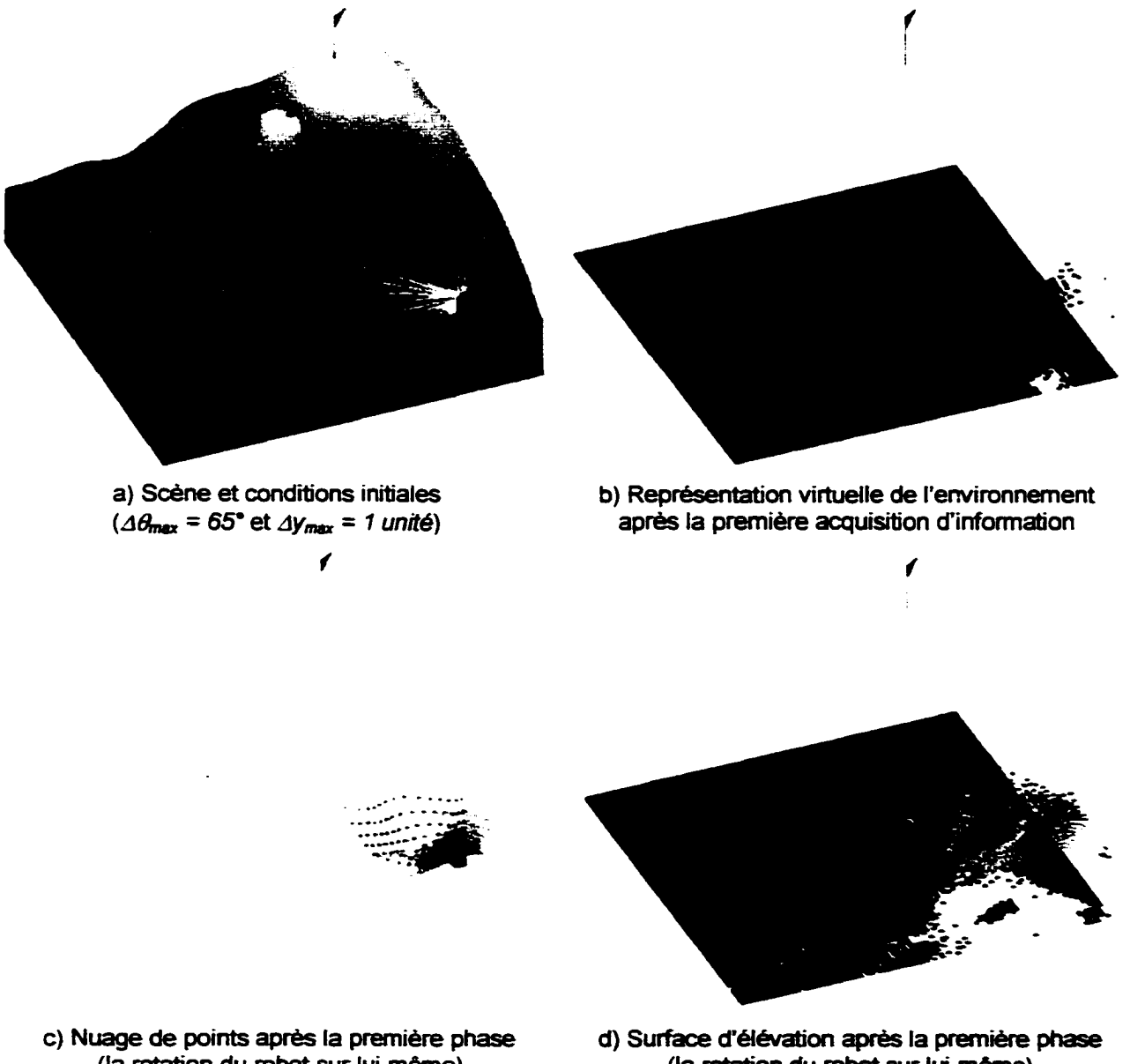
Figure 127 : Chemins praticables

10.7 Exemples de simulation

Cette section présente plusieurs résultats de simulation obtenus en mode autonome. Chaque simulation commence par une courte séquence où le robot pivote de 360° sur lui-même dans le but de connaître son entourage immédiat. Ensuite il suit la trajectoire évaluée à chaque pas. La couleur bleue foncée est utilisée sur les différentes cartes pour indiquer les régions inconnues.

10.7.1 Première simulation

Le premier exemple complet de simulation est présenté en quatre étapes. Chacune des étapes montrent les conditions d'analyse de la scène au début et à chaque tiers du parcours (figures 128 à 131). Cette description exhaustive permet au lecteur de mieux comprendre les détails de l'évolution du robot. Cette simulation totalise 74 505 points.



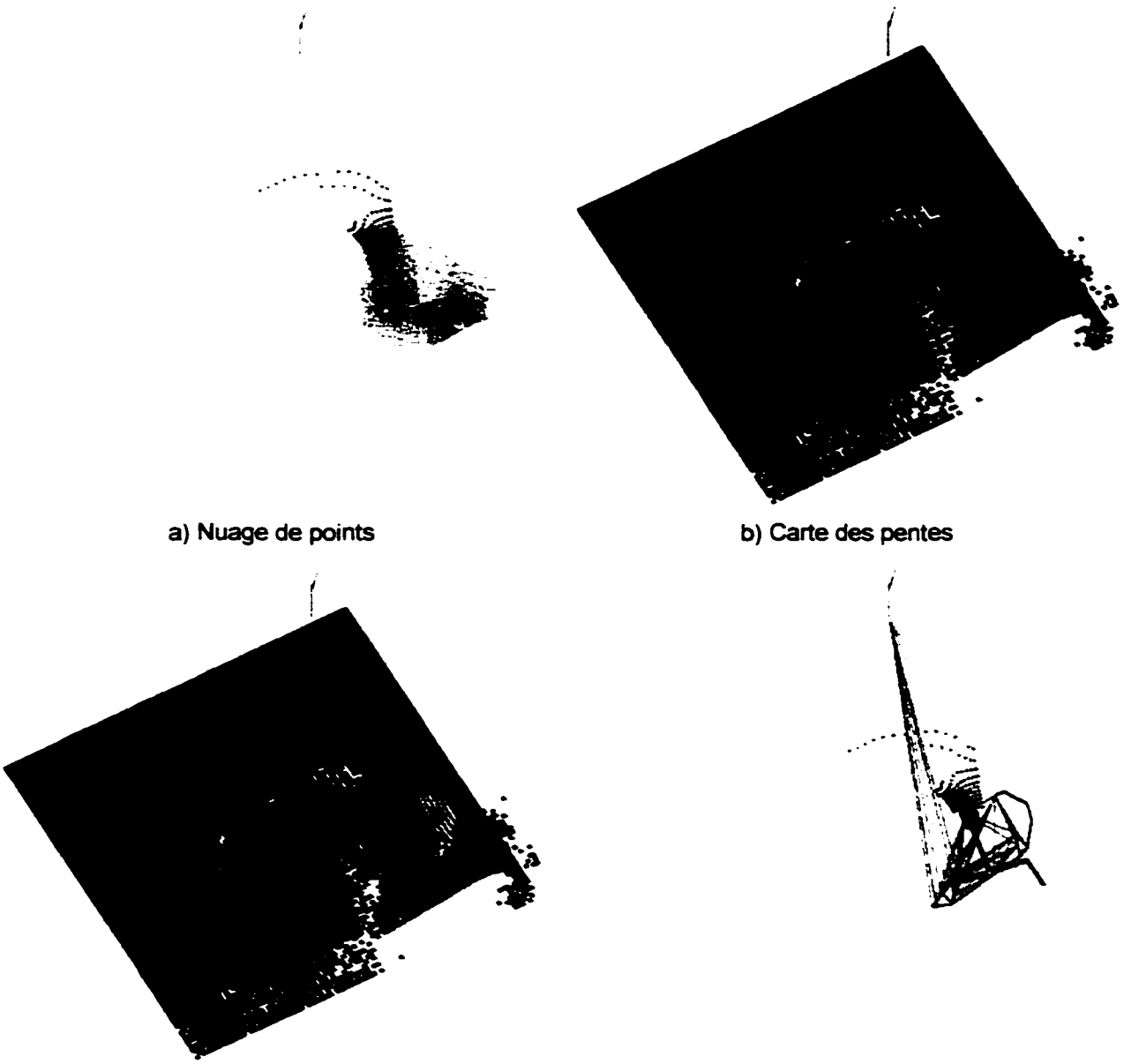
a) Scène et conditions initiales ($\Delta\theta_{max}$ = 65° et Δy_{max} = *1 unité*)

b) Représentation virtuelle de l'environnement après la première acquisition d'information

c) Nuage de points après la première phase (la rotation du robot sur lui-même)

d) Surface d'élévation après la première phase (la rotation du robot sur lui-même)

Figure 128 : Première simulation : état du robot lors de la phase d'initialisation



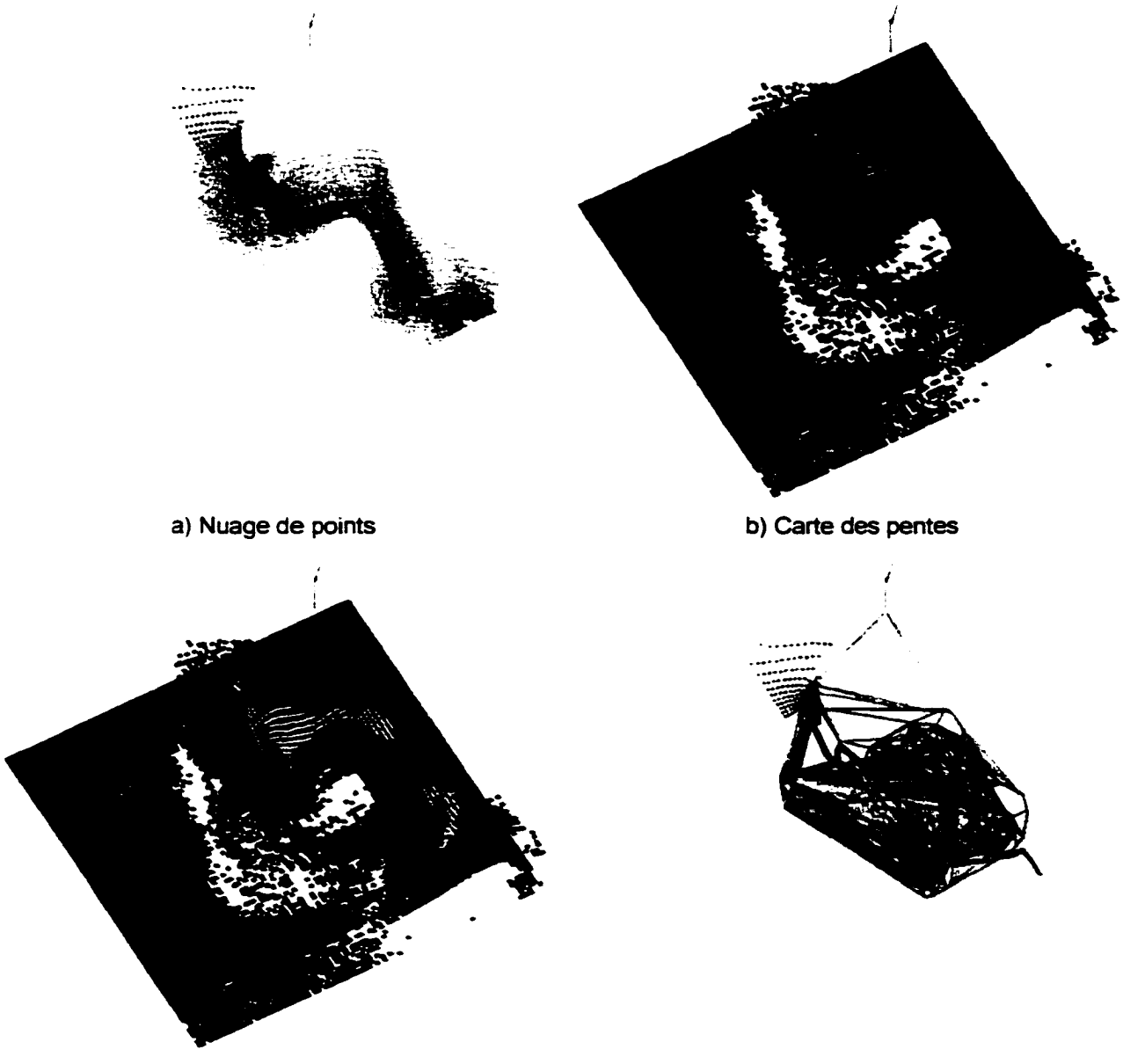
a) Nuage de points

b) Carte des pentes

c) Carte des obstacles

d) Polygones convexes et chemins praticables

Figure 129 : Première simulation : état du robot lors du premier tiers



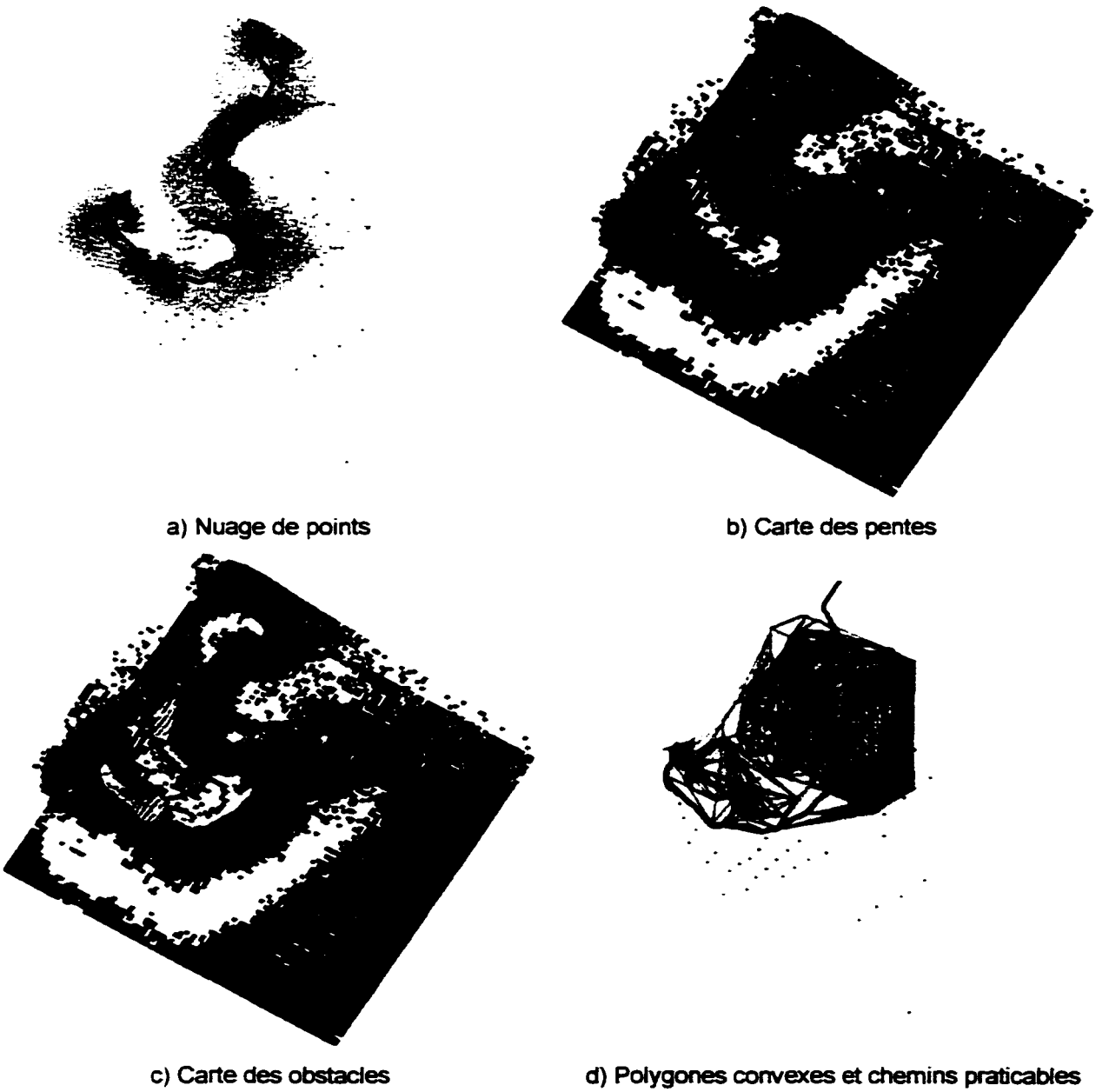
a) Nuage de points

b) Carte des pentes

c) Carte des obstacles

d) Polygones convexes et chemins praticables

Figure 130 : Première simulation : état du robot lors du deuxième tiers



a) Nuage de points

b) Carte des pentes

c) Carte des obstacles

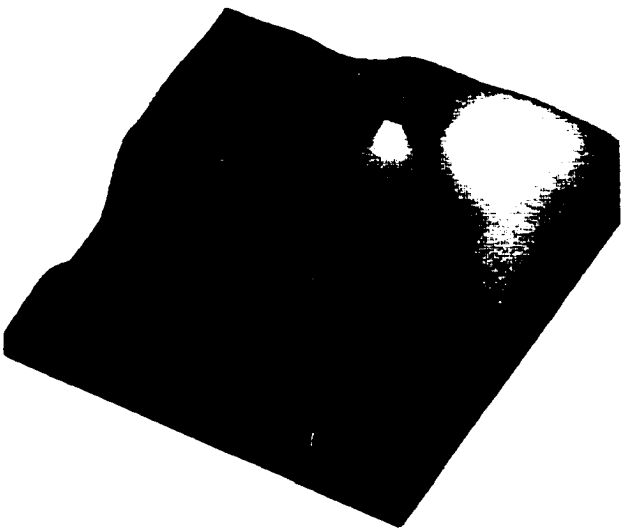
d) Polygones convexes et chemins praticables

Figure 131 : Première simulation : état du robot lors du troisième tiers (lorsque le robot est rendu à destination)

On remarque que le point de vue a changé sur la figure 131 par rapport aux figures précédentes.

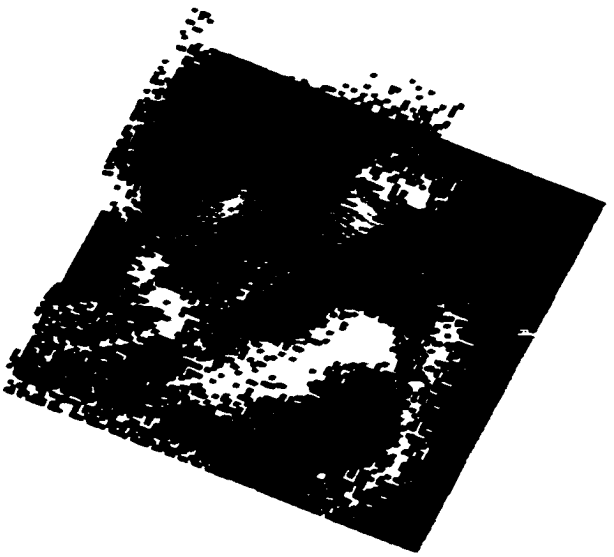
10.7.2 Deuxième simulation

La deuxième simulation présente la même scène mais cette fois avec des positions de départ et d'arrivée différentes. Cette simulation totalise 31 302 points.

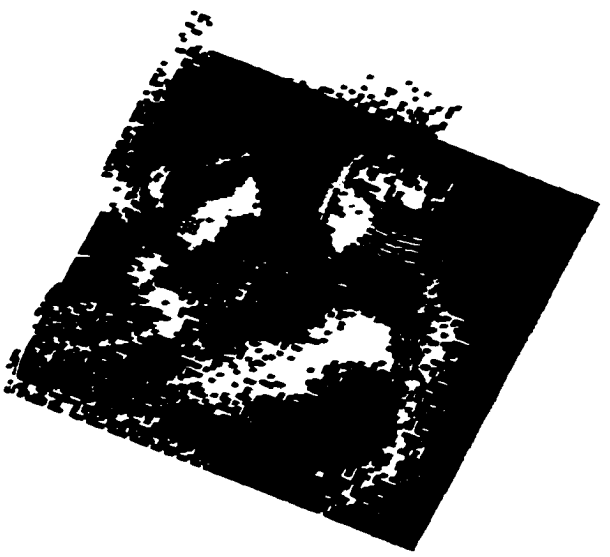


a) Scène et conditions initiales
($\Delta\theta_{max}$ = 65° et Δy_{max} = 1 unité)

b) Nuage de points et trajectoire finale



c) Carte des pentes et trajectoire finale

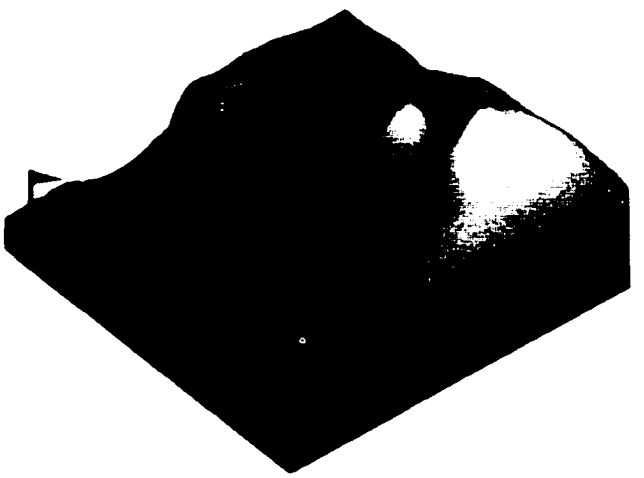


d) Carte des obstacles et trajectoire finale

Figure 132 : Deuxième simulation

10.7.3 Troisième simulation

La troisième simulation présente la même scène mais cette fois avec des positions de départ et d'arrivée différentes. On remarque que le robot est initialement caché derrière une petite montagne. Cette simulation totalise 52 848 points.



a) Scène et conditions initiales
($\Delta\theta_{max}$ = 75° et Δy_{max} = *1 unité*)

b) Nuage de points et trajectoire finale

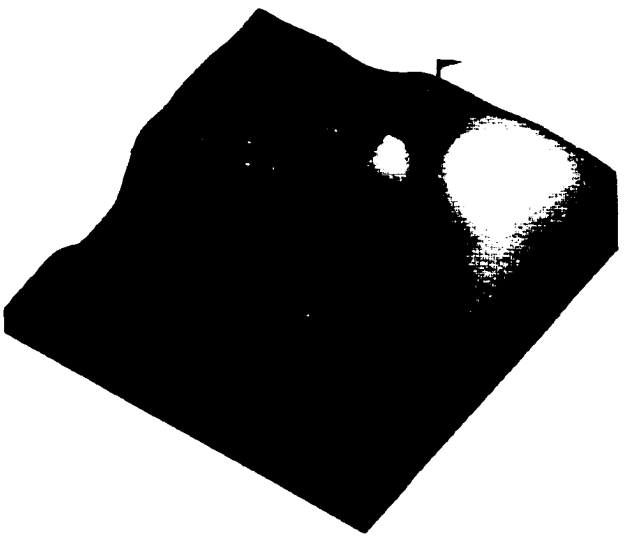
c) Carte des pentes et trajectoire finale

d) Carte des obstacles et trajectoire finale

Figure 133 : Troisième simulation

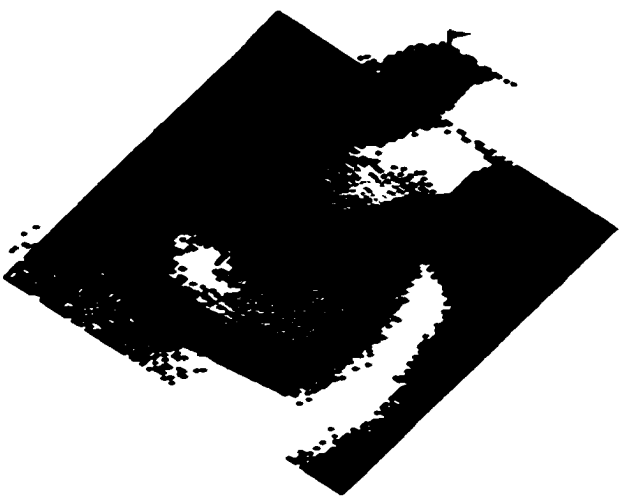
10.7.4 Quatrième simulation

La quatrième simulation présente toujours la même scène et encore une fois des positions de départ et d'arrivée différentes. Cette simulation totalise 87 203 points.

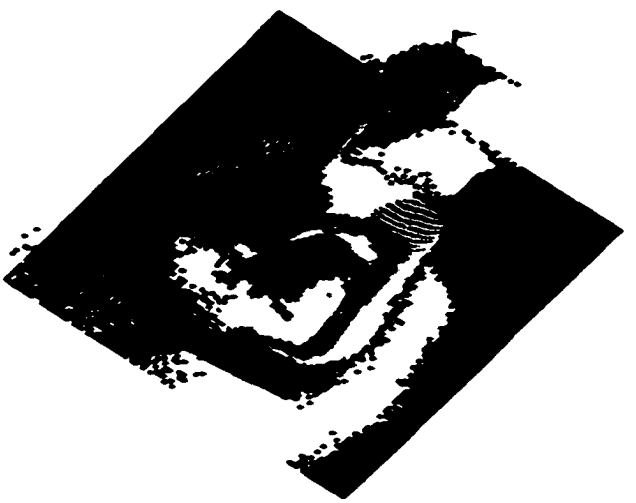


a) Scène et conditions initiales
($\Delta\theta_{max}$ = 55° et Δy_{max} = *1 unité*)

b) Nuage de points et trajectoire finale



c) Carte des pentes et trajectoire finale



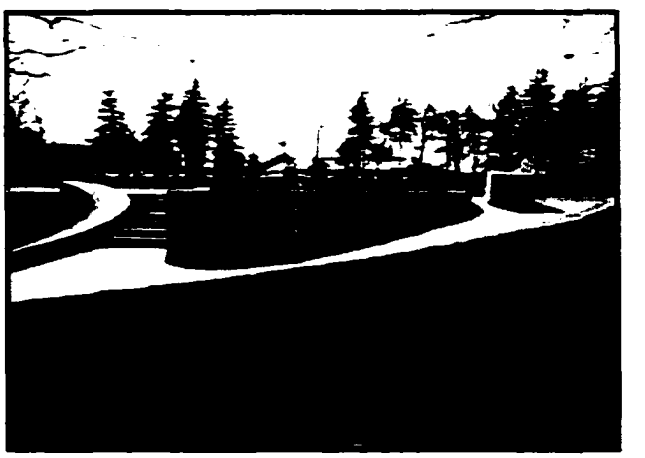
d) Carte des obstacles et trajectoire finale

Figure 134 : Quatrième simulation

10.7.5 Cinquième simulation

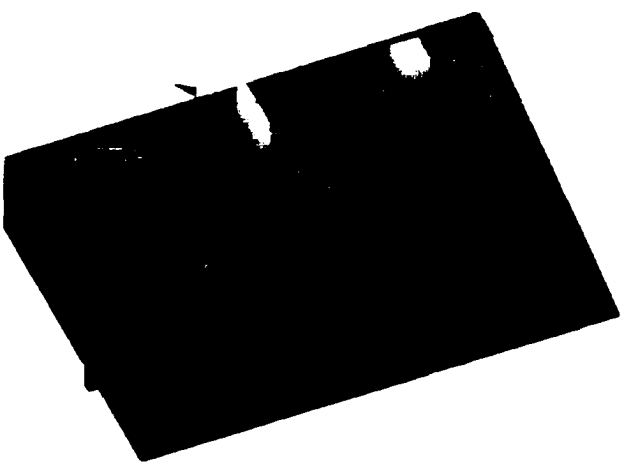
Cette dernière simulation présente une nouvelle scène créé à partir d'un environnement réel. Les deux premières images montrent des photos de la scène réelle. Les deux images suivantes montrent l'image synthétique qui représente la scène et la représentation 3D ainsi créée.

a) Première photo de la scène réelle



b) Deuxième photo de la scène réelle

c) Image synthétique (format Bitmap)



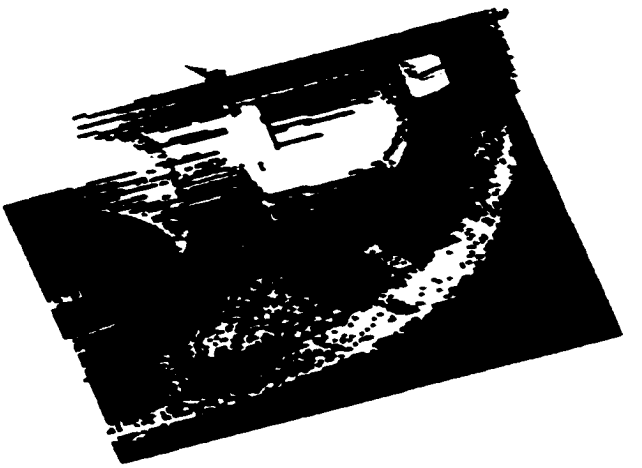
d) Carte synthétique 3D et conditions initiales ($\Delta\theta_{max}$ = 65° et Δy_{max} = 1/4 d'unité)

Figure 135 : Cinquième simulation : scène réelle et définition de la carte réelle

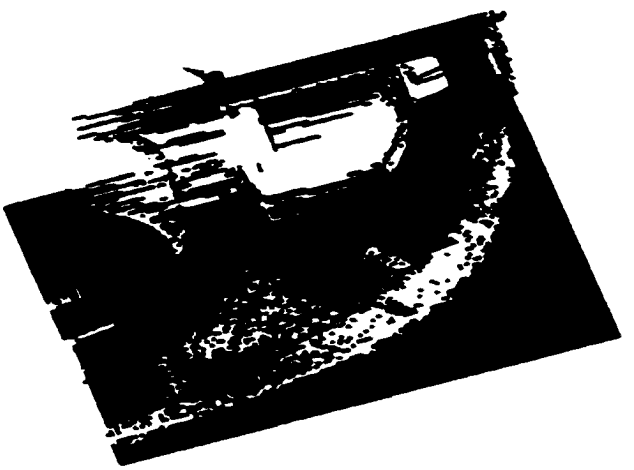
On peut remarquer que cette scène offre deux chemins possibles : un par le biais des escaliers et l'autre par la pente douce. Pour cette mission, les paramètres du robot sont

définis de façon telle que ce dernier ne puisse franchir les escaliers. Cette simulation totalise 90 202 points.

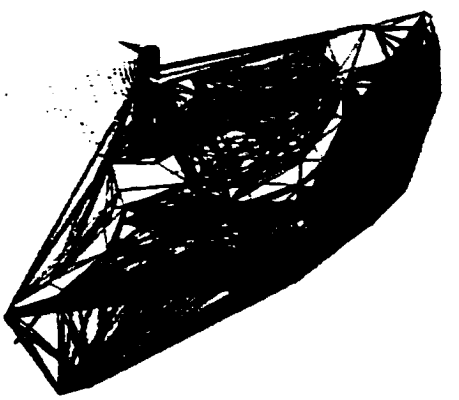
a) Nuage de points et trajectoire finale



b) Carte des pentes et trajectoire finale



c) Carte des obstacles et trajectoire finale



d) Polygones convexes, chemins praticables et trajectoire finale

Figure 136 : Cinquième simulation : Résultat

CONCLUSION ET RECOMMANDATIONS

La tâche de planification de trajectoire est complexe et requiert des algorithmes exigeants en temps de calcul. Cette troisième partie du mémoire présente une brève revue de littérature sur le sujet et explore l'implantation d'une méthode intéressante qui consiste à trouver un chemin passant par les sommets des obstacles décrits par des polygones. On voit que cette méthode considère la distance la plus courte sur la surface 2D½ et la moyenne (ou le maximum) des pentes et des discontinuités verticales à franchir comme critère d'optimisation.

L'une des méthodes de réduction du nombre de sommets des polygones (section 9.3.1) n'est pas optimale car il faut gérer les cas d'exception correspondant aux droites verticales. En utilisant les coordonnées polaires au lieu des coordonnées rectangulaires, on peut éviter ce problème.

Un autre point intéressant à explorer est relié à une planification de trajectoire multirésolution. On commence par une planification de trajectoire globale très approximative et on raffine en augmentant la résolution des informations. Dans le même ordre d'idée, une planification de trajectoire globale jumelée avec une planification de trajectoire locale peut simplifier le problème et augmenter la qualité de la planification.

Il serait intéressant que l'algorithme puisse tenir compte d'autres contraintes importantes telles que la cinématique du robot. La méthode de planification de trajectoire actuelle considère seulement le déplacement du robot sur des segments de lignes droites, ce qui implique qu'à chaque segment de la trajectoire, le robot doit avoir une accélération angulaire infinie pour pouvoir tourner en un temps 0. Pour réaliser cela, le robot doit s'immobiliser, tourner sur place et repartir vers le prochain sommet du chemin optimal.

Une nouvelle approche a été rapidement explorée. Elle a l'avantage de trouver un chemin maximisant la distance aux obstacles. Cette méthode, similaire à la méthode utilisant le diagramme de Voronoi, est décrite par sept étapes :

a. création de faux obstacles de référence servant à modifier la RVE de façon à tenir compte des positions initiale et finale du robot (les obstacles créés sont deux obstacles ponctuels se trouvant de chaque côté de chacune des positions initiale et finale);
b. calcul de la RVE représentant les distances aux obstacles (voir la section 6.1.8);
c. identification des chemins (ces chemins correspondent aux crêtes);
d. validation des chemins en fonction de la taille et de l'orientation du robot;
e. identification des jonctions de chemins (les jonctions de chemin sont les croisements des différentes crêtes);
f. évaluation des coûts pour chaque chemin (en utilisant une méthode semblable à celle décrite à la section 9.7);
g. recherche du chemin le plus court.

Cette nouvelle méthode possède l'avantage de maximiser la distance du robot aux obstacles et ce en tout point. Le temps de calcul de la RVE initial est long mais trouver le chemin optimal est plus court.

Finalement, on présente les résultats obtenus à l'aide d'un simulateur global. Ce dernier simule les résultats obtenus par le système de vision stéréoscopique et implante toutes les techniques présentées dans la deuxième et la troisième partie. Lorsqu'une simulation débute, aucune information n'est connue sur l'environnement. Ensuite, les modules de RVE et de planification de trajectoire permettent de définir les déplacements du robot vers l'objectif à atteindre. À chaque itération, une nouvelle trajectoire est calculée. Ceci permet de tenir compte des dernières informations acquises sur le terrain. Ces simulations montrent que l'approche globale donne des résultats plus que satisfaisants.

CONCLUSION GÉNÉRALE

Ce mémoire présente différents aspects reliés à la robotique mobile : la perception de l'environnement, la représentation virtuelle de l'environnement et la planification de trajectoire. Malgré qu'il reste encore beaucoup de travail à accomplir pour trouver une planification de trajectoire optimale, les objectifs énoncés au début du mémoire sont pleinement atteints.

Le module de perception de l'environnement développé permet de modéliser de façon satisfaisante le relief du terrain se trouvant devant le robot. On désire des mesures fiables, précises et denses du terrain. De plus, le système doit couvrir une superficie assez grande pour que le robot puisse voir les obstacles suffisamment tôt pour s'arrêter. Tous ces objectifs sont réalisés au delà des attentes initiales et les performances obtenues sont des plus encourageantes pour les travaux futurs.

Quand à lui, le module de représentation virtuelle de l'environnement permet de fusionner et de synthétiser le nuage de points initial en une nouvelle représentation consistant en deux cartes. La première représentation, sous forme de surface d'élévation, permet au module de contrôle de la démarche de connaître les endroits où le robot peut poser ses pieds. La deuxième représentation, sous la forme de la carte des obstacles, permet au module de planification de trajectoire de trouver un chemin qui évite les obstacles.

Ensuite, le module de planification de trajectoire est présenté par l'approche du plus court chemin suivant les contours d'obstacles polygonaux. Cette méthode permet de considérer les critères du chemin le plus court sur la surface 2D½ et des moyennes (ou des maximums) des pentes et des discontinuités verticales sur le chemin.

Les simulations effectuées ont permis de comprendre en détail toutes les étapes du processus et de valider l'approche globale. On est maintenant en mesure d'affirmer que l'approche globale est fonctionnelle et qu'elle est prometteuse.

Finalement, on réalise que ces trois aspects du problème touchant à la robotique mobile ne sont pas simples à résoudre et nécessitent des efforts de conception et de développement considérables. Malgré tout, l'enjeu en vaut la peine car les robots de demain seront l'outil ultime tant convoité.

RÉFÉRENCES

Agarwal, P. K., Biedl, T. et Lazard, S. (1998) Curvature-Constrained Shortest Paths in a Convex Polygon. *Proceeding 14th Annual Symposium on Computational Geometry, June 7-10, 1998, Minneapolis, MN, AMC Press, pp 392-401*

Alt, H. et Yap, C. K. (1990) Algorithmic Aspects of Motion Planning : A Tutorial. *Algorithms Review, May 1990, pp 61-77*

Ayache, N. (1989) *Vision stéréoscopique et perception multisensorielle, application à la robotique mobile.* Paris : InterEditions, 341 p.

Bicchi, A., Casalino, G. et Santilli, C. (1995) Planning Shortest Bounded-Curvature Paths for a Class of Nonholonomic Vehicles among Obstacles. *IEEE International Conference on Robotics and Automation.*

Boissonnat, J.-D. et Yvinec, M. (1998) *Algorithmic Geometry.* Cambridge University Press, 519 p.

Bouroche, J.-M. et Saporta, G. (1980) *L'analyse des données.* Presse Universitaire de France, 127 p.

Castellanos, J. A. et Tardos, J D. (1999) *Mobile Robot localization and Map Building, A Multisensor Fusion Approach.* Kluwer Academic Publisher, 205 p.

Chavand, F. et Colle, E. (1998) *Perception de l'environnement en robotique.* Hermes, 191 p.

Cherif, M. (1999) Motion Planning for All-Terrain Vehicles : A Physical Modeling Approach for Coping with Dynamic and Contact Interaction Constraints. *IEEE Transactions on Robotics and Automation, Vol. 15, No. 2, April, 1999*

Cunha, S. R. et Coimbra de Matos, A. C. (1994) A Global Motion Control Strategy Using Dynamic Programming. *Oceans 94 Osates, Brest, France, 1994*

de Berg, M., van Kreveld, M., Overmars, M. et Schwarzkopf, O. (1998) *Computational Geometry : Algorithms and Applications.* Springer, 365 p.

DeBlois, S., Lambert, G., Lessard, P., Lupien, S. et Sankar, T. (2000) Design of Capra – A Quadruped Robot With Improved Agility. *International Symposium on robotics, May 2000, Montréal, p. 392-297*

Dijkstra, E. W. (1959) A Note on two Problems in connexion with graphs. *Numerische Mathematik 1, pp 269-271*

Dudek, G et Jenkin, M. (2000) *Computationnal principles of mobile robotics.* Cambridge University Press, 280 p.

Donald, B., Xavier, P., Canny, J. et Reif, J. (1993) Kinodynamic Motion Planning. *Journal of the A CM, vol. 40(5), pp. 1048-1066*

Everett, H. R. (1995) *Sensors for Mobile Robots : Theory and application.* A. K. Peters, Ltd, 528 p.

Falkenhagen, L. (1995). Depth Estimation from Stereoscopic Image Pairs Assuming Piecewise Continuous Surfaces. *Institut für Theoretische Nachrichtentechnik und Informationsverarbeitung Universität Hannover, 14 p.*

Farritor, S. et Dubowsky, S. (1997) A Self-Planning Methodologiy for Planetary Robotics Explorers. *8th International Conference on Advanced Robotics, pp. 449-504, Monterey, CA, July 7-9, 1997*

Faugeras, O. D. et Toscani G. (1986) The calibration problem for stereo. *In proceedings CVPR '86, IEEE, Miami beach, États-Unis, pp 15-20*

Fournier, M. (1999) Numérisation en temps réel du relief 3D au sol devant le robot Capra à partir d'une caméra 2D et d'un émetteur laser. *École de technologie supérieure, Projet de session pour le cours Vision par ordinateur (SYS-844), 41 p.*

Geiger, D., Ladendorf, B. et Yuille, A. (1994) Occlusions and Binocular Stereo. *42 p.*

Guldner, J. et Utkin, V. (1993) Sliding Mode Control for an Obstacle Avoidance Strategy Based on an Harmonic Potential Field. *Proceeding of the 32th Conference on Decision and Control, San Antonio, Texas, December 1993*

Horaud, R et Monga, O. (1993) *Vision par ordinateur : Outils fondamentaux.* Hermes, 378 p.

Howard, A. et Kitchen, L. (1998) Navigation using natural landmarks. *Robotics and Autonomous Systems, No. 26, 1999, pp. 99-115*

Janet, J. A., Luo, R. C. et Kay, M. G. (1997) Autonomous Mobile Robot Global Motion Planning and Geometric Beacon Collection Using Traversability Vectors. *IEEE, Transaction on Robotics and Automation, Vol. 13, No. 1, Feb. 1997, pp. 132-139*

Lagarde, B. (1999) Étude et conception d'un cou robotique stabilisateur d'images. *École de technologie supérieure, 29 p.*

Lambert, G. (1999) Modélisation mathématique et conception mécanique du robot marcheur/sauteur Capra. *École de technologie supérieure, 1999, 39 p.*

Latombe, J.-C. (1991) *Robot motion planning.* Kluwer Academic Publishers, 651 p.

Lazanas, A. et Latombe, J.-C. (1995) Motion Planning with Uncertainty : A Landmark Approach. *Artificial Intelligence, 76(1-2), pp. 285-317, 1995*

Lebart, M., Morineau, A. et Piron, M. (2000) *Statistique exploratoire multidimensionnelle.* Dunod, 439 p.

Lessard, P. (2002) *Contrôle de la démarche de Headus – Robot quadrupède dynamique.* École de technologie supérieure, 169 p.

Lupien, S., Lessard, P. Demers, J.-C., Lambert, G., Bigras, P. et Sankar, S. (2001) Versatile Robot Platform For Hazardous Environment and Automation Applications. *Proceedings of the 2001 ASME Design Engineering Technical Conference, Pittsburgh, P.A. Sept. 2001*

Lupien, S. (2001) Développement d'outils d'aide à la conception de robots marcheurs. *École de technologie supérieure, 123 p.*

Magherbi, D. et Wolovich, W. A. (1992) Real-Time Velocity Feedback Obstacle Avoidance via Complex Variables and Conformal Mapping. *Proceeding of the 1992 IEEE, International Conference on Robotic and Automation, Nice, France, May 1992*

Mantegh, I., Jenkin, M. R. M. et Goldenberg, A. A. (1997) Solving the Find-Path Problem : A complete and less complexe approach using the BIE methodology. *IEEE 1997*

Moravec, H. (2002) *Robust Navigation by Probabilistic Volumetric Sensing.* [En ligne]. *http://www.frc.ri.cmu.edu/~hpm/* (Consulté le 1 juin 2001)

Otsu, N. (1979) A Threshold Selection Method from Gray-Level Histogram. *IEEE Transaction on Systems, Man and Cybernetics, Vol. 9, No. 1, pp. 62-66*

Pagnot, R. et Grandjean, P. (1995) Fast Cross-Country Navigation on Fair Terrains. *IEEE International Conference on Robotics and Automation.*

Pai, D. K. et Reissell, L.-M. (1998) Multiresolution Rough Terrain Motion Planning. *IEEE Transactions on Robotics and Automation, Vol. 14, No. 1, February 1998*

Preparata, F. P. et Shamos, M. I. (1985) *Computational Geometry : An Introduction.* Springer-Verlag, 398 p.

Pruski, A. (1996) *Robotique mobile : la planification de trajectoire.* Hermes, 236 p.

Shiller, Z. et Gwo, Y.-R. (1991) Dynamic Motion Planning of Autonomous Vehicles. *IEEE Transaction on Robotics and Automation, Vol. 7, No. 2, April, 1991*

Shiller, Z. (1999) Motion Planning for Mars Rover. *First workshop on Robot Motion and Control, Kiekrz, Poland, June 28, 1999*

Siméon, T. et Dacre-Wright, B. (1993) A Practical Motion Planner for All-terrain Mobile Robots. *Proceedings of the 1993 IEEE, International Conference on Intelligent Robots and Systems, Yokohama, Japan, July 26-30, 1993*

Sugihara, K. et Smith, J. (1997) Genetic Algorithms for Adaptive Motion Planning of an Autonomous Mobile Robot. *Proceedings of the 1997 IEEE International Symposium on Cumputational Intelligence in Robotics and Automation (CIRA '97), June 10-11, 1997*

Tomasi, C. et Manduchi, R. (1996) Stereo Without Search. *European Conference on Computer Vision (ECCV), 13 p.*

Trucco, E. et Verri, A. (1998) *Introductory techniques for 3-D computer vision.* Etats-Unis : Prentice-Hall, 341 p.

Tsai. R. Y. (1986) An efficient and accurate camera calibration technique for 3D machine vision. *Proc. IEEE Conf. Computer Vision and Pattern Recognition. pp. 364-374.*

Voisin. Y.. Marzani. F. et Diou A. Calibration d'un système de vision par lumière structurée. *Vision Interface (VI). May 14. 8 p., 2000*

Withesides. S. H. (1985) Computational Geometry and Motion Planning. *School of Cumputer Science. McGill University. Technical Reports SOCS-85.14. April 1985*

ANNEXE 1 : INFORMATION SUR LE PROJET CAPRA

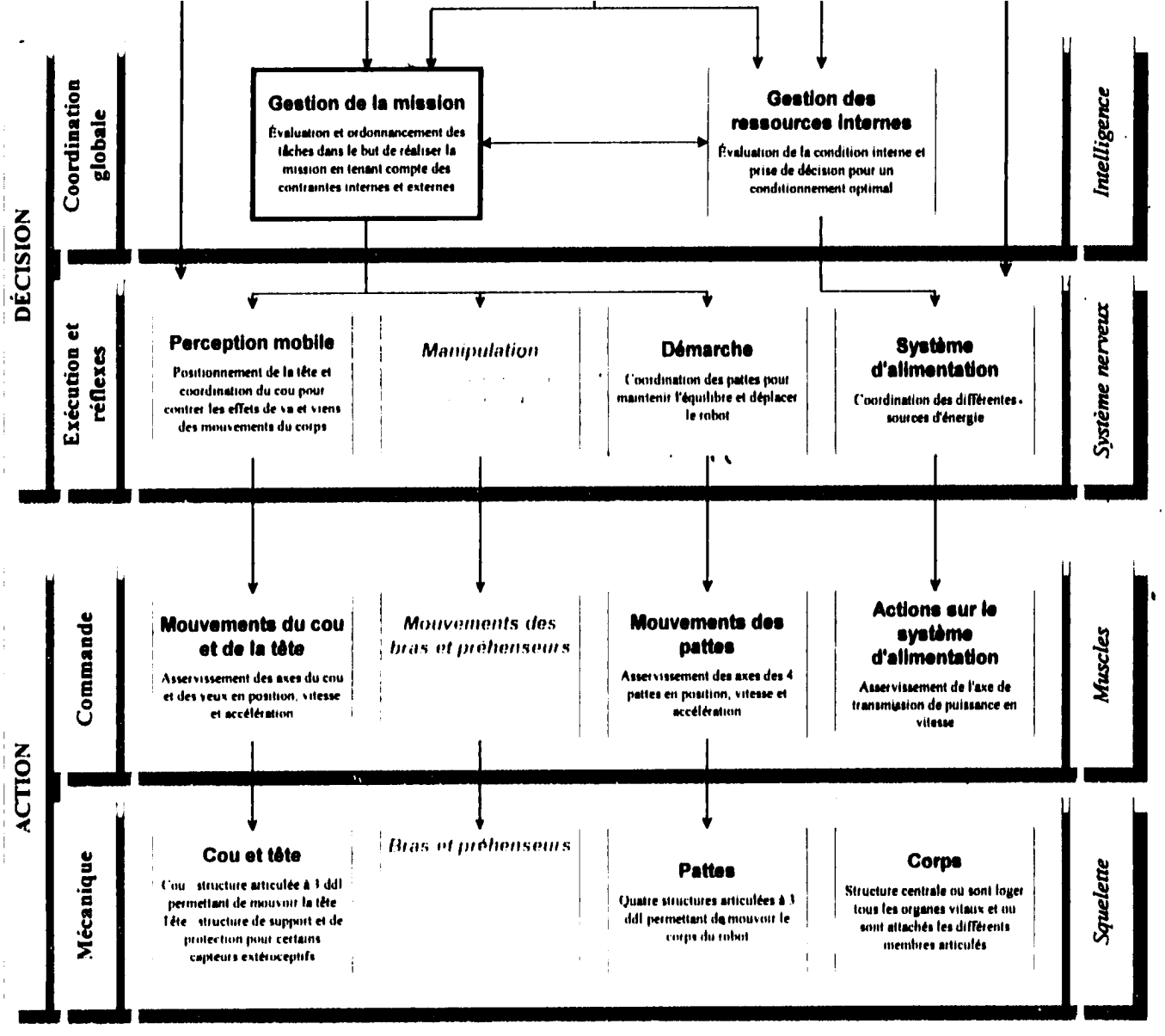
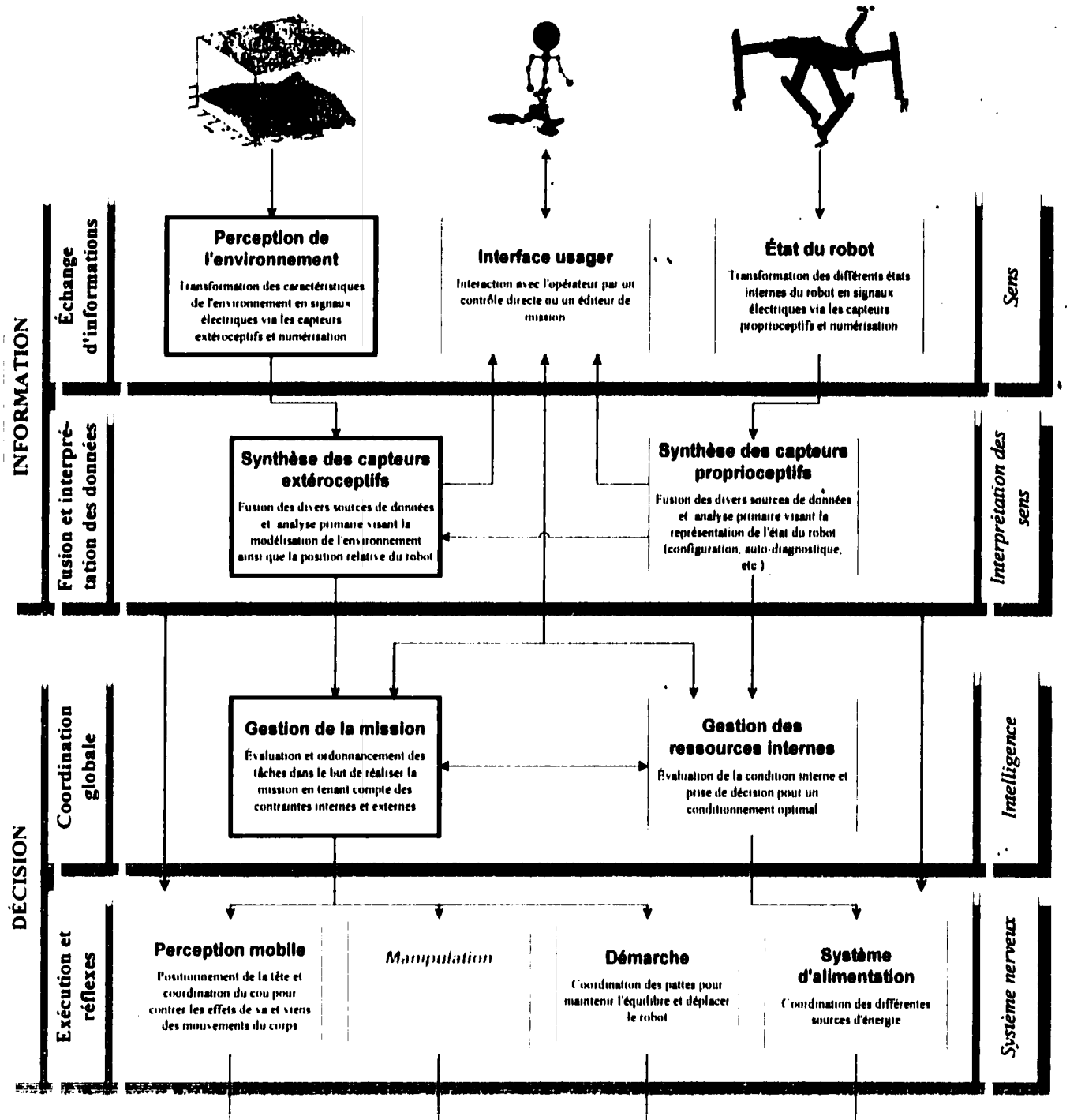


Figure 137 : Structure conceptuelle de Capra


INFORMATION
DÉCISION
Échange d'informations
Fusion et interprétation des données
Coordination globale
Exécution et réflexes
Sens
Interprétation des sens
Intelligence
Système nerveux
Perception de l'environnement
Transformation des caractéristiques de l'environnement en signaux électriques via les capteurs extéroceptifs et numérisation
Interface usager
Interaction avec l'opérateur par un contrôle directe ou un éditeur de mission
État du robot
Transformation des différents états internes du robot en signaux électriques via les capteurs proprioceptifs et numérisation
Synthèse des capteurs extéroceptifs
Fusion des divers sources de données et analyse primaire visant la modélisation de l'environnement ainsi que la position relative du robot
Synthèse des capteurs proprioceptifs
Fusion des divers sources de données et analyse primaire visant la représentation de l'état du robot (configuration, auto-diagnostique, etc.)
Gestion de la mission
Évaluation et ordonnancement des tâches dans le but de réaliser la mission en tenant compte des contraintes internes et externes
Gestion des ressources internes
Évaluation de la condition interne et prise de décision pour un conditionnement optimal
Perception mobile
Positionnement de la tête et coordination du cou pour contrer les effets de va et vient des mouvements du corps
Manipulation
Démarche
Coordination des pattes pour maintenir l'équilibre et déplacer le robot
Système d'alimentation
Coordination des différentes sources d'énergie