Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi-structuré

Juan Gabriel Avina Cervantes

Institut National
Polytechnique de Toulouse

E.N.S.E.E.I.H.T.

École doctorale d'informatique et télécommunications

Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi-structuré

THÈSE

présentée et soutenue publiquement le 15 Février 2005

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Toulouse
(spécialité signal, image et acoustique)

par

Juan Gabriel AVIÑA CERVANTES

Composition du jury

Président :	Raja Chatila	Directeur de recherche CNRS, LAAS-CNRS
Rapporteurs :	Christine Fernandez-Maloigne	Professeur à l'université de Poitiers
	Roland Chapuis	Professeur au CUST à Clermont Ferrand
Examinateur :	Michel Cattoën	Professeur à l'INP de Toulouse
Directeur de thèse :	Michel Devy	Directeur de recherche CNRS, LAAS-CNRS

Laboratoire d'Analyse et d'Architecture des Systèmes

Avant-propos

Ce manuscrit présente l'ensemble des travaux de recherche que j'ai effectué au cours des trois années de ma thèse dans l´équipe Robotique et Intelligence Artificielle au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS à Toulouse.

Je tiens à remercier en tout premier lieu Michel Devy qui a dirigé cette thèse dans la continuité de mon stage de D.E.A. Tout au long de ces trois années, il a su orienter mes recherches aux bons moments. Pour tout cela, et pour sa confiance, son soutien et la liberté d'action qu'il m'a accordée, je le remercie vivement.

Je remercie les directeurs successifs du LAAS, Jean-Claude Laprie et Malik Ghallab de m'avoir accueilli dans leur établissement.

J'adresse également mes remerciements à Monsieur Raja Chatila, Directeur de Recherche CNRS et responsable du groupe RIA, pour m'avoir permis de travailler au sein de son groupe et pour avoir accepté de participer à mon jury de thèse.

Je remercie les rapporteurs de cette thèse Christine Fernandez-Maloigne, Professeur à l'Université de Poitiers et Roland Chapuis, Professeur à l'Université Blaise Pascal de Clermont-Ferrand pour la rapidité et la pertinence avec laquelle ils ont lu mon manuscrit et l'intérêt qu'ils ont porté à mon travail. Merci également à Michel Cattoën, Professeur à l'Institut National Polytechnique de Toulouse, qui a accepté de juger ce travail, en tant qu'examinateur dans le jury.

Parmi ceux qui ont contribué à mes réflexions, je remercie tout spécialement Thierry Peynot, Aurelie Clodic, Andres Restrepo, Joel González, Antonio Marín, Ignacio Herrera, Leonardo Solaque, Claudia Esteves, Jean-Bernard Hayet, Sylvain Argentieri et Frédéric Lerasle.

Enfin, une pensée émue pour tous les étudiants avec qui j'ai partagé une salle, un café, un repas ou une console d'ordinateur pendant ces trois années : Gustavo Arechevaleta, Diana Mateus,... et toute la troupe très conviviale de RIA.

Enfin, je ne saurais terminer ces remerciements sans mentionner les proches, famille et amis, qui m'ont soutenu par leurs encouragements. Je remercie particulièrement mes enfants, Eli et Pénélope, et ma femme, Paula, pour m'avoir suivi pendant ce long cycle d'étude et pour leur soutien moral pendant la rédaction du manuscrit.

Je dédie cette thèse
à ma famille : Paula Dalida,
Eli Gabriel et Pénélope Juliette. [1]

[1] I suddenly see the solution of a puzzle-picture. Before there were branches there ; now there is a human shape. My visual impression has changed and now I recognize that it has not only shape and colour but also a quite particular 'organization'

Table des matières

Liste des figures

Introduction

La motivation première des travaux présentés dans cette thèse, est le développement de techniques visuelles appropriées pour l'automatisation de machines autonomes devant intervenir dans des environnements extérieurs. Nous participons en particulier, à un projet du LAFMI[2], dédié à la robotique agricole : les robots pourraient se charger des récoltes ou de la cueillette, de la fertilisation ou fumigation des champs, du transport des aliments... La figure 1, présente une machine entièrement réalisée par l'université mexicaine partenaire de ce projet[3] : le but est de concevoir un système embarqué capable de piloter automatiquement cette machine.

FIG. 1 – Machine agricole à équiper d'un pilote automatique

Les tâches d'une telle machine se dérouleraient dans des environnements extérieurs semi-structurés dans lesquels, malgré l'absence ou la pauvreté des informations géométriques (*e.g.*, pas de lignes droites, de marquage routier, de surfaces planes...), le robot peut reconnaître au cours de ses déplacements, une certaine structure introduite par l'intervention de l'homme : chemins, haies, fossés, arbres... Nous considérons que les chemins sont les régions qui vont relier tous les buts d'intérêt dans un milieu agricole. Ainsi, les primitives élémentaires de déplacement

[2]LAboratoire Franco-Mexicain en Informatique

[3]FIMEE : Facultad de Ingeniería Mecanica Électrica y Electrónica (Université de Guanajuato)

du type « Suivre le chemin » ont une grande importance dans la conception de tâches beaucoup plus complexes.

Comme le robot doit exécuter des tâches en milieu extérieur, cela impose plusieurs contraintes qui doivent être surmontées ou au moins atténuées : aucun contrôle sur l'illumination, conditions météorologiques variables, grande variabilité des scènes selon le temps, la saison, le lieu..., vibrations dans les systèmes d'acquisition (*e.g.*, images floues), transmission de données limitée ou impossible vers une station centrale, *etc.*

Dans nos travaux, nous considérons que la machine ne dispose que d'un seul capteur : une caméra couleur.

En utilisant la vision couleur, un robot peut réaliser des tâches complexes telles que le contrôle des moteurs pour exécuter des déplacements, l'évitement d'obstacles, la localisation par rapport à des amers naturels, le suivi d'objets dynamiques... Ainsi, le domaine d'application du robot devient plus vaste et varié, mais en même temps l'intégration de ces fonctionnalités vont multiplier les problèmes de synchronisation de données, de fusion entre données acquises à des instants différents, de compacité des représentations construites avec ces données et de l'exploitation de ces représentations pour planifier et exécuter des déplacements ou des tâches.

Plusieurs thèses seraient nécessaires pour traiter de tous ces sujets. Nous nous sommes intéressés (1) à l'extraction d'une description $2D$ qui doit informer sur les objets présents dans la scène courante et sur leur interdépendance, puis (2) à la reconnaissance et au suivi dynamique des objets naturels donnés dans ces descriptions.

De telles fonctions peuvent être appliquées sans modifications majeures, à l'extraction des chemins de terre ou à l'identification des fruits ayant une couleur ou une texture spécifiques. Parmi les traitements que le robot doit exécuter pour se déplacer sur ce chemin, ou pour saisir ce fruit, le traitement des images est certainement l'opération la plus coûteuse en temps de calcul, la plus délicate vu la complexité de l'environnement et la plus critique, car si la vision échoue, la tâche échoue. Ceci nous impose de réaliser un système le plus robuste possible ; nous avons en particulier étudié les méthodes destinées à maintenir la même perception visuelle de la couleur des objets en dépit des changements des conditions d'illumination.

Sous le vocable *Navigation visuelle*, nous désignons toutes les fonctions qu'un système autonome doit exécuter pour planifier et exécuter ses déplacements dans un environnement quelconque, inconnu a priori, perçu par une ou plusieurs caméras embarquées. Nos travaux sont dédiés à la navigation visuelle pour un robot équipé d'une seule caméra couleur.

Parmi les fonctions que le robot doit savoir exécuter, citons :

(1) l'acquisition d'une image et l'extraction d'une description 2D de la scène courante ; il s'agit d'une fonction d'interprétation, qui identifie dans l'image, les entités de l'environnement utiles pour la navigation d'un robot terrestre : zones de terrain navigables, obstacles, chemins, amers...

(2) la construction d'un modèle de l'environnement par fusion des descriptions acquises depuis les positions successives du robot.

(3) la planification d'une trajectoire requise par une tâche que doit exécuter le robot, typiquement dans notre contexte « Aller jusqu'au champ A », « Aller à la ferme » ... Cette trajectoire est générée à partir du modèle de l'environnement, acquis au préalable par le robot, ou introduit par un opérateur dans la base de connaissances du système.

(4) enfin, l'exécution de trajectoires, par séquencement de mouvements élémentaires contrôlés à partir des images.

Nos contributions portent sur les fonctions visuelles utiles à la navigation. Tout d'abord, exploiter avec une certaine robustesse, des images couleur en extérieur, donc sans aucun contrôle sur l'illumination sur la scène, requiert plusieurs pré-traitements effectués sur l'image : nous avons analysé l'état de l'art en ce domaine, et nous proposons une méthode d'adaptation chromatique exécutée en ligne, afin d'assurer une relative invariance de l'image aux variations d'illumination.

Notre contribution principale porte sur l'extraction automatique d'une description $2D$ de la scène acquise depuis une image couleur, description qui contient les régions navigables (en particulier, les *chemins*) : nous nous sommes inspirés dans nos travaux, des résultats préliminaires obtenus en 1998 au LAAS par R.Murrieta Cid [Murrieta-Cid 98]. Cette description est exploitée à plusieurs fins, en particulier afin d'exécuter des commandes référencées vision, comme *Suivre un chemin*.

Cette fonction d'analyse de la scène courante, est intégrée sur notre démonstrateur DALA[4] dans un module appelé *R-LOC*.

Afin de contrôler l'exécution d'une commande référencée vision comme *Suivre un chemin*, le système visuel doit fournir en temps réel, la position des entités utiles à la navigation (région navigable, objet à rejoindre...). Dans le cadre de la vision active, les méthodes de suivi d'objets (*tracking*) ont des limitations inhérentes à leur initialisation et à leur possible dérive qui peut entraîner une mauvaise localisation dans l'image, de l'entité suivie ; en ce domaine, nous avons collaboré avec A. Marin Hernandez [Marin-Hernandez 04] qui a travaillé sur les méthodes de suivi visuel, fondées en particulier sur les contours actifs. Notre contribution porte sur l'initialisation et la détection de dérive du suivi, à partir d'images couleur.

La fonction de *tracking* est intégrée sur DALA, dans un module appelé *R-TRACK*.

Nous avons entamé des travaux sur la modélisation de l'environnement dans lequel le robot doit se déplacer. Comme nous nous limitons à modéliser un réseau de chemins, nous proposons de construire un modèle topologique. Notre contribution porte sur la détection et la catégorisation des intersections ou carrefours entre chemins. Vu la très grande variabilité des environnements, nos résultats restent modestes : nous saurons résoudre le problème dans des cas simples (allées dans une pelouse), mais traiter le cas général (intersections en rase campagne, dans une forêt, carrefour entre chemins de terre et routes ...), pourrait faire l'objet d'une autre thèse...

Enfin, en collaboration avec D.Mateus [Mateus 05], nous avons validé l'ensemble de nos travaux, par des expérimentations qui exploitent les modules de vision *R-LOC* et *R-TRACK* intégrés sur le robot mobile DALA. Les informations visuelles sont utilisées pour réaliser des primitives de mouvement *Suivre un chemin* ou *Aller vers objet*, tout en évitant les obstacles.

Notons que l'implémentation de ces techniques dans un robot ayant des ressources de mémoire limitées, impose des contraintes importantes : les algorithmes développés doivent s'exécuter à mémoire constante et en temps réel (typiquement, vu la vitesse lente de nos robots, mieux que 1Hz pour toute la chaîne de traitement).

Le présent rapport s'articule en cinq chapitres. Le premier présente un panorama général des contributions les plus importantes proposées dans la littérature, dans le domaine de la naviga-

[4]Robot mobile tout terrain du groupe de Robotique et Intelligence Artificielle du LAAS/CNRS

tion visuelle pour un robot mobile. Nous détaillerons plus particulièrement les travaux dans des environnements d'extérieur semi-structurés, plutôt dédiés aux véhicules autonomes ou à l'aide à la conduite de véhicules. Nous faisons aussi un bref rappel des travaux menés par le groupe de recherche *Robotique et Intelligence Artificielle* du LAAS, sur la navigation dans les environnements naturels. Enfin, nous introduirons ainsi le schéma général de la méthode de navigation que nous avons développée.

Le deuxième chapitre est consacré à l'acquisition d'images couleur, depuis la caméra jusqu'à l'obtention d'images exploitables pour les traitements suivants ; notre propos est illustré par les résultats obtenus sur notre robot DALA sur lequel est montée une caméra mono-CCD dotée d'un filtre *Bayer*. Étant donné que l'analyse des scènes d'extérieur depuis les images acquises depuis le robot, se fonde sur la couleur, la caractérisation et le rendu correct de ces images s'avèrent essentiels. La *quête* des traiteurs d'images couleur est l'invariance de la couleur des objets perçus dans la scène, en fonction de la chaîne d'acquisition et des conditions d'illumination. Nous présentons les améliorations apportées dans le rendu d'images couleur à partir de l'étude des algorithmes de demosaïquage, la correction « on-line » de la balance des blancs et la correction *gamma*. Ces méthodes permettent de reproduire des images couleur de haute résolution et d'obtenir une relative *constance des couleurs*.

Le chapitre 3 présente la procédure qui nous permet d'extraire la description $2D$ de la scène. Nous présentons d'abord une méthode hybride de segmentation des images couleur qui nous permet d'extraire les principales régions de l'image, correspondant aux entités présentes dans la scène. Les régions segmentées sont caractérisées par des attributs de texture et de couleur et par des attributs contextuels. Enfin, une méthode de classification permet d'identifier les éléments courants dans la scène (chemin, herbe, arbre, ciel, champ labouré...). Pour cela, nous avons essayé deux approches en mode supervisé : les *Support Vector Machines* (SVM) et les k plus proches voisins (k-PPV). Une réduction des informations redondantes dans la base de données par analyse en composantes indépendantes (ACI) permet d'améliorer le taux global de reconnaissance des régions.

Dans le chapitre 4, l'identification des régions *Chemin* est exploitée pour acquérir le modèle du réseau dans lequel le robot va devoir se déplacer. Les contours des chemins détectés dans une image sont généralement très bruités ; ils sont d'abord lissés par des courbes de Bézier. Dans un réseau de chemins, le robot doit reconnaître les intersections de chemins lui permettant (a) dans une phase d'apprentissage, de construire un modèle topologique du réseau et (b) dans une phase de navigation, de planifier et exécuter une trajectoire topologique définie dans ce réseau. Ainsi, la méthode de mise en correspondance de formes « Shape Context » est évaluée pour traiter de la catégorisation de chemins (carrefour, intersection, impasse *etc.*), cela afin de construire un graphe topologique. Nous avons d'abord validé la méthode de catégorisation des formes de chemins, sur des images réelles acquises en campagne ; puis, nous présentons des résultats préliminaires sur la construction d'un modèle topologique sur une séquence d'images de synthèse.

Finalement, le dernier chapitre présente des résultats expérimentaux obtenus sur le démonstrateur DALA. Une trajectoire que doit suivre le robot, est exprimée par une séquence de primitives de déplacement élémentaires *Suivre un chemin*, *Aller vers objet*, *Suivre une bordure*... ; une telle primitive exploite l'information fournie par le système de vision présenté dans les chapitres précédents. Nous décrivons la primitive *Suivre un chemin* : sur chaque image acquise par la caméra embarquée, le robot détecte la position courante du chemin dans l'image, construit

une trajectoire au sol (avec l'hypothèse de terrain plat) pour avancer sur le chemin en restant en son milieu et en évitant les obstacles, filtre cette trajectoire en la fusionnant avec celles obtenues depuis les images précédentes et lance l'exécution de la trajectoire fusionnée.

Étant donné que le modèle sémantique de la scène est produit à basse fréquence (de 0,5 à 1 Hz) par le module de vision couleur, nous avons intégré avec celui-ci, un module de suivi temporel des bords du chemin dans la séquence d'images. Le suivi est traité par une méthode de contour actif (*Snakes*), adaptée pour des images couleur. Ce suivi permet d'augmenter la fréquence d'envoi des consignes (de 5 à 10 Hz) au module de locomotion. Modules de vision couleur et de suivi temporel doivent être synchronisés de sorte que le suivi puisse être réinitialisé en cas de dérive : nous évoquons ainsi, sans prétendre les résoudre, les problèmes difficiles liés à l'intégration sur un système embarqué, de plusieurs processus visuels.

Une conclusion termine ce mémoire de thèse en faisant une récapitulation des principaux résultats obtenus ainsi que les perspectives de recherche dans cette thématique très difficile, de la navigation autonome de robot mobile en milieu extérieur semi-structuré à partir de la vision.

Chapitre 1

Contexte de la navigation visuelle

1.1 Introduction

Les études présentées dans ce mémoire de thèse sont consacrées à la problématique de la navigation visuelle d'un robot de façon autonome et robuste, dans des environnements naturels semi-structurés. Plus précisément, nous nous sommes intéressés à l'étude du déplacement du robot dans les milieux agricoles en percevant visuellement les principaux objets qui l'entourent.

Étant donné que les scènes naturelles manquent de structure, une carte numérique ou métrique de l'environnement est généralement très compliquée à obtenir ; du fait des caractéristiques de ces environnements (terrain non plat, peu de plans ou de segments de droite...), la construction d'une carte est beaucoup plus complexe que pour des environnements intérieurs. Le robot doit donc se servir de la vision pour détecter et classifier les zones propices à son déplacement (chemins, terrains plats, terrains d'herbe courte ...) mais aussi pour détecter d'autres objets (arbres, roches, bâtiments ...) qui, soit pourraient l'empêcher d'avancer, soit pourraient être utiles pour accomplir sa tâche (par exemple, aller vers un arbre pour cueillir des fruits).

Bien que la reconnaissance visuelle d'objets puisse effectuer des fausses détections, la connaissance globale de la scène va nous donner une information contextuelle très utile, y compris pour détecter et corriger des erreurs d'interprétation. Une telle connaissance de l'environnement s'avère essentielle car l'information structurée est faible et limitée en milieu extérieur. Ainsi, dans notre application, le *système de vision* du robot fournit la majorité des informations exploitables pour calculer les consignes à transmettre au module de *locomotion* [Avina-Cervantes 03a] : ce sont les contours délimitant les régions navigables, indiquant un objet vers lequel le robot doit aller, ou encore indiquant la présence d'obstacles qui pourraient empêcher le déplacement du robot. Ces approches de recherche de repères visuels pour guider les mouvements du robot, nécessitent la sélection et la reconnaissance de différents objets ou configurations exploités ensuite pour calculer des localisations relatives, à la fois métriques et topologiques.

Les mouvements de notre robot se limitent à des actions simples : longer un mur, suivre un chemin, aller vers un objet, éviter un obstacle. Des applications futures bénéficieraient cependant de systèmes véritablement autonomes, capables de se déplacer en terrain (partiellement) inconnu ou dynamique, grâce à la capacité du robot à reconnaître contextuellement son environnement.

Sur notre robot, le système de vision est constitué par une tête stéréo constituée de caméras Micropix C-1024. Ce modèle de caméras a la particularité d'avoir un seul capteur CCD[5] équipé

[5]Le « Charged Coupled Device » fut disponible en 1974, basé sur silicium, qui présente une réponse quasiment linéaire en fonction de l'intensité lumineuse

d'un filtre ou mosaïque Bayer [Gunturk 04] qui permet d'obtenir des images couleur de haute qualité (résolution maximale 1024×768).

Dans cette thèse, pour obtenir la description de la scène depuis chaque image, en particulier, pour extraire les zones navigables de l'environnement, une seule caméra est utilisée.

La navigation visuelle d'une machine autonome, ou d'un robot mobile en milieu extérieur semi-structuré, n'est pas très différente de l'évolution d'un véhicule dit autonome (AGV[6]) [Miura 97, Miura 02] pour la conduite assistée. Les deux problématiques sont liées et les méthodes qui mènent à leur solution le sont aussi. Il existe déjà des prototypes d'applications robotiques pour l'agriculture (récolte de fruit [Fernandez-Maloigne 93], culture de la terre, application sélective des herbicides [Lee 99]) mais leur précision et vitesse sont encore au dessous des attentes.

La section suivante présente un état de l'art très général de la navigation visuelle, puis nous placerons nos travaux de recherche dans le vaste contexte de la navigation visuelle des robots autonomes en milieu naturel (par exemple, Mars rovers) ou de véhicules autonomes en milieu routier. Enfin nous allons illustrer de manière globale l'approche de navigation proposée.

1.2 Vision pour la navigation de robots mobiles

Chez l'humain, la vision est le sens qui transmet le plus d'informations au cerveau. Elle nous fournit une grande quantité de données en provenance de l'environnement et nous permet d'entreprendre une interaction intelligente avec les *environnements dynamiques* (évitement d'obstacles mobiles, rendez-vous avec autres agents mobiles...). De ce fait, il n'est pas surprenant de trouver une grande quantité de recherches sur le développement de capteurs qui essaient d'imiter le système visuel humain [Siegwart 04]. De plus, les capteurs visuels utilisés par les *robots intelligents* doivent avoir les mêmes sensibilités et réponses à la lumière que notre système de vision.

En robotique, au cours des deux dernières décennies, les innovations technologiques concernant la fabrication de caméras et l'évolution des ordinateurs ont permis d'intégrer des systèmes complexes de vision dans les systèmes embarqués, que ce soit sur des robots mobiles pour la navigation autonome ou sur des véhicules pour l'aide à la conduite. La vision artificielle revêt une importance toute particulière car elle permet de fournir à la machine les capacités nécessaires pour réagir avec son environnement ; elle fournit les représentations à partir desquelles le robot prend des décisions.

Pour décrire l'état de l'art dans le domaine de la navigation de robots mobiles, nous faisons une distinction basée sur la structuration de l'environnement :

- les environnements *s*tructurés peuvent être représentés par des primitives géométriques simples (*i.e.*, détection de lignes droites, surfaces planes, couloirs, portes, ...).
- les environnements *n*on structurés sont considérés pour des applications en milieu vraiment naturel (site planétaire, polaire, forestier...) ; en milieu terrestre, ce sont de riches sources d'information contextuelle, de couleur et de texture.
- les environnements *s*emi-structurés sont en essence des environnements naturels qui ont subi une modification partielle de l'homme, typiquement les sentiers ou chemins laissés par des passages fréquents de l'homme ou des animaux, par exemple dans le cadre d'activités agricoles.

Plus précisément, les contributions scientifiques de la navigation visuelle, sont classifiées en deux catégories : la première, la plus prolifique, pour robots d'intérieur et la seconde, en pleine

[6] Automated Guided Vehicle

croissance, pour robots d'extérieur. Les avancées sur ces deux fronts, intérieur et extérieur, ont été significatives. Par exemple, il y a vingt ans, il était difficile d'imaginer qu'un robot d'intérieur pût trouver son chemin dans une salle ou un couloir encombrés ; ceci n'est plus considéré comme un défi [Hayet 02, Kosaka 92]. De nos jours, seulement pour citer l'un des nombreux projets, le projet FINALE[7] permet aux robots de dépasser des vitesses de 17 m/min en utilisant une architecture d'ordinateur (Pentium II, 450 MHz) sans matériel spécial pour le traitement des signaux. Le robot utilise ses capteurs ultrasons pour éviter des obstacles mobiles et stationnaires [Kosaka 92, Pan 98], il utilise en même temps la vision pour se localiser.

On trouve également des progrès de la vision dans la robotique d'extérieur structuré, *e.g.*, les systèmes NAVLAB [Thorpe 88, Aufrere 03], le suivi de routes et chemins guidé par des attributs visuels [Turk 88, Dickmanns 88, Meng 93, Rasmussen 02], les résultats obtenus dans le cadre du projet EUREKA *Prometheus* qui a essayé d'améliorer la circulation et la sécurité du trafic sur route [Regensburger 94], parmi d'autres approches [Belluta 00, Talukder 02, Dickmanns 99].

L'université de Carnegie Mellon (CMU) a développé un ensemble de systèmes de navigation automatique pour la route. Parmi ces systèmes, les plus connus sont : RALPH [Pomerleau 96] (Rapidly Adapting Lateral Position Handler), ALVINN [Baluja 96] (Autonomous Land Vehicle in a Neural Network), AURORA [Chen 95] (Automotive Run-Off Road Avoidance) et ALVINN-VC [Jochem 95b] (VC comme acronyme de *V*irtual Camera).

Sans doute, le succès le plus médiatique des systèmes NAVLAB (Navigation Laboratory) a été le test de conduite « No hands across America » qui consistait en un trajet de 2849 miles de Pittsburgh, en Pennsylvanie à San Diego, en Californie. Au cours de ce trajet, le véhicule NAVLAB 5 [Jochem 95a] a été piloté de manière autonome pendant 98,2% du parcours.

En Europe, concernant la robotique en milieu extérieur non structuré, d'autres travaux ont été effectués au Centre National d'Études Spatiales à Toulouse. Le robot de fabrication russe Marsokhod, équipé d'un système de navigation basé sur la stéréovision a permis la validation de plusieurs études sur la robotique d'exploration planétaire. Les équipes réunies au sein du réseau VIRGO [Paletta 00] (Vision-based Robot Navigation Network), ont déjà lancé quelques machines automatisées surprenantes basées sur le concept de *vision active et intentionnelle* (exploration des aspects de l'environnement qui lui sont nécessaires à un moment donné). Le réseau VIRGO a pour objectif la coordination de la recherche européenne dans le développement des systèmes robotiques intelligents dotés de la capacité de navigation sur des environnements partiellement inconnus ou dynamiques.

En outre, les chercheurs du Centre de Systèmes Autonomes du Royal Institute of Technologies de Stockholm ont conçu un robot muni d'une caméra centrale et de deux caméras latérales. Grâce aux informations fournies par ce système de vision, le robot peut se déplacer et éviter les obstacles en utilisant un minimum de commandes motrices. Un autre exemple est le robot mis au point à l'Institute of Computer Science, en Grèce, qui planifie ses déplacements en « visualisant » certains éléments invariants d'un espace (des affiches, des extincteurs, des signalisations murales, des lumières au plafond, des portes, *etc.*) et en utilisant ensuite ces repères pour orienter sa navigation.

Certains prototypes expérimentaux présentent, en outre, des comportements « intelligents ». Ils sont capables d'agir ou de réagir en fonction à la fois des conditions locales de l'environnement et de leur objectif de navigation. Par exemple, le robot Sir Arthur, mis au point au Centre national allemand de recherche en technologies de l'information, est doté de six pattes dont les mouvements sont commandés par seize moteurs. La machine a commencé à apprendre,

[7] acronyme de Fast Indoor Navigation Allowing for Locational Errors

par « essais-erreurs », la séquence de mouvements de chacun de ses membres lui permettant d'avancer en ligne droite. Ensuite, elle a été dotée d'un détecteur de lumière, afin d'accomplir un travail plus difficile : rejoindre dans l'espace qui l'entourait, l'endroit bénéficiant de l'éclairage le plus intense.

Enfin, l'environnement réel d'un robot ne se limite pas à des éléments immobiles. En France notamment, l'INRIA travaille sur des méthodes de détection des mouvements d'objets dont les contours ne sont pas clairement définis. C'est une tâche particulièrement complexe, indispensable à de nombreux usages potentiels de la robotique.

(a) Dala, le véhicule tout-terrain.

(b) Lama, ROVER pour l'exploration planétaire.

Fig. 1.1 – Les robots mobiles du LAAS pour la navigation dans des environnements naturels.

Au LAAS, la recherche en robotique mobile dans des environnements naturels est menée dans le cadre du projet EDEN[8] par le groupe de Robotique et Intelligence Artificielle [Chatila 98, Mallet 00, Lacroix 02, Gonzalez 02]. Afin d'atteindre les objectifs du projet, le groupe s'est doté d'outils pour la conception, le développement et l'intégration de l'algorithmique dans des modules fonctionnels, composants de base dans une architecture décisionnelle temps réel. Les premières applications de ce projet concernaient l'exploration planétaire, la navigation autonome [Betge-Brezetz 96a, Betgé-Brezetz 96b] et la modélisation de l'environnement, soit $2D$ [Murrieta-Cid 98, Murrieta-Cid 02] (régions navigables, sur un terrain quasiment plat), soit $3D$ (en utilisant une carte d'élevation, une carte de régions ou une carte d'objets $3D$ [Parra-Rodriguez 99]). Désormais, le projet a évolué avec le même esprit vers la coopération entre les robots aériens (Karma) et terrestres (Dala, Lama).

Concernant la robotique d'intérieur, le robot mobile Rackham déployé à la Cité de l'Espace à Toulouse, est équipé de caméras et d'un télémètre laser. Il navigue de manière autonome dans l'environnement de l'exposition ; il dispose également de fonctions et de dispositifs permettant l'interaction avec les visiteurs.

Dans les deux sections suivantes, nous présentons avec plus de détails les avancées de la vision pour la navigation en milieu intérieur, puis extérieur.

[8] Expérimentations de Déplacements en Environnement Naturel, projet interne au groupe RIA du LAAS

1.3 Navigation visuelle en milieu intérieur

Dès les premières projets en robotique mobile, les séquences d'images ont été proposées pour fournir des informations utiles à la navigation d'un robot [Giralt 79]. Elles sont généralement traitées par des méthodes de reconnaissance de formes pour détecter une cible connue par un modèle (apparence de la cible ou modèle géométrique d'un objet) dans les images successives. La plupart de ces approches utilisent des modèles structurés ; le processus de navigation est associé à l'une des modalités suivantes :

- *utilisation de cartes connues de l'environnement :* le système dépend, pour se déplacer et se localiser, d'une carte (modèle) géométrique, probabiliste ou topologique de son environnement. Elle est fournie par l'utilisateur [Hayet 02]. Mais ces modèles ne sont pas toujours faciles à obtenir. Les systèmes de vision permettent au robot de se localiser dans ces modèles, en détectant certains amers de l'environnement.
- *construction incrémentale d'une carte au fur et à mesure des déplacements :* le robot construit son propre modèle de l'environnement en utilisant l'information provenant des différents capteurs (ultrasons, caméras, lasers, *etc.*) pendant une phase d'exploration [Thrun 98]. Dans un premier temps, le robot doit acquérir un modèle adapté pour sa propre localisation ; ce sont souvent des cartes stochastiques éparses contenant les représentations paramétriques des amers détectés par le robot tandis qu'il se déplace dans l'environnement. Ces cartes sont construites de manière incrémentale grâce à des techniques d'estimation (filtrage de Kalman, filtrage particulaire...) proposées (1) pour localiser le robot dans l'environnement et (2) pour fusionner les données acquises depuis la position courante. Ces méthodes, très étudiées depuis une quinzaine d'années, traitent donc du problème connu sous le mnémonique SLAM, pour *Simultaneous Localization And Mapping.*
  Une fois qu'il a acquis une carte d'amers, le robot sait se localiser ; il peut alors acquérir d'autres représentations, typiquement un modèle de l'espace libre.
- *navigation dépourvue de carte :* dans ce type de navigation, le robot ne se sert pas d'une représentation explicite de son environnement, mais plutôt de connaissances plus qualitatives ou topologiques, construites à partir de la reconnaissance d'objets immergés dans la scène ou de la détection et du suivi temporel de cibles visuelles [Marin-Hernandez 04] (extincteurs, bureaux, amers plans, corridors, *etc.*). Citons aussi des approches bio-inspirées, comme par exemple une méthode fondée sur le flux optique, inspirée par le système visuel des abeilles [Rizzi 98].

En définitive, il s'avère que les approches géométriques sont bien adaptées aux environnements d'intérieur et des modèles mathématiques formels sont souvent proposés dans la littérature. Ces modèles contiennent implicitement des actions pour éviter les obstacles, pour la détection d'amers, la construction ou l'actualisation de cartes et l'estimation de la position du robot.

1.4 Navigation visuelle en milieu extérieur

Pour les milieux naturels d'extérieur, la construction d'une carte est nettement plus compliquée (niveau de structuration faible : extraction plus complexe de primitives géométriques), à plus forte raison quand les scènes changent dynamiquement (à cause de la météo, de la saison, des conditions d'illumination, *etc.*) ou quand d'autres agents dynamiques partagent le même environnement (piétons, autres robots, véhicules,...). Ce type de navigation peut être divisé en deux classes suivant le type de structuration, *i.e.*, navigation en environnements structurés ou

non structurés.

Étant donné que nous utilisons à la fois des éléments naturels (sols ou terrains plats) et des éléments artificiels produits par l'homme dans son parcours (chemins), notre robot évolue dans un environnement *dynamique semi-structuré d'extérieur*.

1.4.1 Environnement d'extérieur structuré

La navigation en environnement structuré d'extérieur concerne les techniques de suivi de routes [Chapuis 95] ou de chemins en terrain plat). Dans l'état de l'art actuel, le succès du suivi de routes repose sur la bonne détection et estimation des marquages (lignes blanches) et sur leur interprétation pour établir sur quelle voie navigable se trouve le véhicule [Aufrère 01]. Dans ces systèmes, les modèles de l'environnement ne sont pas exagérément complexes grâce à l'utilisation d'informations telles que les points de fuite (*vanishing points*) [Rasmussen 04a, Rasmussen 04b], ou autres propriétés géométriques de la route ou de la voie.

Par analogie, la navigation sur des routes est similaire à la navigation dans des couloirs pour les environnements d'intérieur à l'exception des problèmes causés par les ombrages, des conditions d'illumination changeantes et du phénomène de la constance chromatique [Rosenberg 01]. Sadayuki Tsugawa [Tsugawa 94] a mis au point l'un des premiers systèmes autonomes en utilisant un système de caméras stéréo placé verticalement pour détecter et éviter des obstacles sur la route. Le système de Tsugawa (daté de la deuxième moitié des années 70) ne dépassait pas les 30 km/h.

En France, Roland Chapuis [Chapuis 95, Aufrère 00] a développé un algorithme performant de localisation et de suivi de route en temps réel pour l'aide à la conduite (en détectant les lignes blanches). Un système comprenant un ordinateur portable et une caméra vidéo située au niveau du rétroviseur intérieur a été embarqué sur le véhicule expérimental VELAC[9] du LASMEA [Aufrère 01] et testé avec succès sur plusieurs dizaines de kilomètres de routes ou d'autoroutes jusqu'à une vitesse de 100 km/h. Cette même approche s'est également avérée robuste dans le cadre de la détection de routes ou de chemins goudronnés dépourvus de marquages ou lignes blanches [Aufrere 04].

Par ailleurs, la méthodologie d'extraction de chemins a été adaptée aussi au milieu sous-marin car la détection de chemins de terre n'est pas si différente de la détection des pipelines sous l'eau. Ainsi, l'INRIA a mis au point un système pour le contrôle [Malis 99] d'un ROV (Remotely Operated Vehicle) sous-marin durant une tâche d'inspection visuelle de pipeline [Rives 97].

Pour le suivi de chemins sans marquages, citons le module VITS [Turk 88] (Vision Task Sequencer), intégré au véhicule Alvin (Autonomous Land Vehicle) qui a été mis au point par l'agence aérospatiale Martin Marietta à Denver, Colorado. VITS est un ensemble général de routines pour le suivi de chemins, de détection et d'évitement des obstacles. Dans ce système, une caméra couleur CCD (images RGB de 480x512) était montée sur une platine qui est contrôlée en site et azimut par un sous-système de vision.

Le module de vision fournissait une description du chemin en considérant que le véhicule était devant la zone navigable. Cette description était complétée par des informations telles que la position, la vélocité, le cap, *etc.* , pour mieux localiser le chemin en cherchant uniquement dans une zone de l'image. Enfin, un modèle global de la scène était obtenu. Le bon fonctionnement du processus est vérifié par l'évaluation successive des modèles de la scène calculés sur

[9] Véhicule Expérimental du LASMEA pour l'Aide à la Conduite

les images précédentes, cela afin d'assurer le lissage et la continuité des contours du chemin. La segmentation était l'opération basique pour dégager les pixels appartenant au chemin. Évidemment, les défis consistaient à surmonter les problèmes des ombres d'arbre, la luminosité du soleil, les flaques d'eau *etc.* Un algorithme de seuillage (*thresholding*) dynamique identifiait les pixels du chemin en projetant d'abord dans le repère du monde la frontière du chemin obtenu précédemment et en rétro-projetant ensuite un trapèze représentant la zone où rechercher le chemin sur l'image courante.

Également, le laboratoire de navigation NAVLAB au CMU a développé depuis 1984 des véhicules contrôlés par ordinateur pour des applications de conduite assistée ou automatique [Jochem 95b]. Son dernier modèle le NAVLAB 11 est une jeep robotisée équipée d'une grande variété de capteurs pour la détection d'obstacles de moyenne et courte portée (lidars, lasers, radars, sonars) ainsi que des caméras panoramiques [Thorpe 03]. Le premier prototype, le NAVLAB 1, utilisait la vision couleur pour le suivi du chemin et l'information $3D$ pour la détection et l'évitement d'obstacles [Thorpe 88]. Les composantes couleur R, G, B étaient stockées à différentes résolutions dans une pyramide d'images. Les images de la plus haute résolution servaient à quantifier la texture des régions dans la scène tandis que celles de basse résolution fournissaient l'information chromatique. La classification de pixels dans l'image (chemin ou non chemin) dépendait donc d'une relation probabiliste sur les attributs de couleur et de texture. L'une des dernières étapes consistait à appliquer une transformée de Hough pour obtenir les paramètres de la route : l'orientation et le point de fuite (*vanishing point*) [Rasmussen 04a]. Ceci permettait de corriger la classification des pixels de l'image actuelle et de prédire la position de la route dans la prochaine.

Le véhicule NAVLAB 1 fut aussi équipé d'un système de navigation, basé sur un réseau de neurones (ALVINN) sur lequel l'apprentissage utilisait un algorithme de rétropropagation. L'idée de base du système ALVINN consistait à observer le comportement du conducteur humain à intervalles de temps réguliers et « d'apprendre » ses réactions sur la route [Pomerleau 96]. En utilisant ce module, la vitesse maximale fut de 20 miles/h sur le prototype NAVLAB 1, puis de 55 miles/h sur un véhicule amélioré : le NAVLAB 2. Une version plus performante du ALVINN, l'ALVINN-VC (Virtual Camera) permet une meilleure détection de la route et même des intersections [Jochem 95b, Jochem 95a] ; ce système exploite des « images virtuelles » pour fournir des points de vues différents de celui de la vraie caméra, vues qui seraient acquises à plusieurs distances de la position actuelle de la vraie caméra.

Le projet EUREKA, *Prometheus* [Dickmanns 88, Dickmanns 99] cherchait à profiter du potentiel de la vision robotique pour améliorer la sécurité dans la circulation de véhicules, notamment en allégeant la fatigue provoquée par la conduite monotone [Regensburger 94]. L'objectif a été de développer un copilote artificiel qui préviendrait le conducteur du danger dans des situations de fatigue, mais qui pourrait aussi reprendre le contrôle du véhicule dans des situations extrêmes. Cette approche utilisait une méthode d'extraction d'attributs par corrélation contrôlée, ce qui permettait une implémentation en temps réel et des vitesses de 96 km/h en autoroutes et de 40 Km/h sur chemins goudronnés sans marquage de lignes. Quelques démonstrations en public ont été menées en 1994 sur l'autoroute A1 près de Paris [Dickmanns 02].

Pour surmonter les contraintes de temps, d'autres approches préfèrent s'appuyer sur des modèles de vision active en visant uniquement l'objet ou la région d'intérêt [Kelly 98] ; de telles méthodes exploitent implicitement des connaissances a priori sur l'environnement. Dans la section suivante nous présentons les travaux de navigation visuelle les plus pertinents, adaptés aux

environnement non structurés.

1.4.2 Environnement d'extérieur non structuré

Les environnements non structurés manquent de primitives géométriques régulières (lignes blanches bien délimitées, largeur de la voie relativement constante, *etc.*) qui sont exploitées dans les approches décrites ci-dessus pour effectuer la navigation. Ce type de milieu contient plutôt des chemins de terre à la campagne, des terrains accidentés ou n'importe quelle zone traversable par un véhicule (robot) tout-terrain. C'est dans cette catégorie que se situent tous les projets dédiés à l'exploration planétaire [Wilcox 92, Chatila 95a] utilisant des ROVERs avec locomotion tout-terrain (comme le démonstrateur LAMA) et avec un niveau élevé d'autonomie.

Dans les applications *terrestres*, le robot exécute généralement une tâche prédéfinie, comme le suivi d'un élément qu'il doit reconnaître dans l'environnement tout au long d'une région navigable [Lorigo 97]. La détection des zones navigables [Betge-Brezetz 96a] et la détection d'obstacles lors du déplacement du robot [Mallet 00] exploitent la construction et la fusion des cartes de traversabilité. Pour pouvoir exécuter une tâche de navigation, le robot requiert des fonctions additionnelles pour la détection et le suivi d'amers (discontinuité sur la ligne d'horizon, bâtiments, un grand arbre ...) exploités pour se localiser ou pour exécuter des commandes asservies.

C'est ici que la vision (stéréo ou monoculaire) joue son rôle par des techniques de segmentation, de caractérisation/classification par texture et couleur dans l'image segmentée, méthodes les plus adaptées pour maintenir le véhicule sur la région[10] navigable ; cependant, rares sont les papiers qui rapportent l'utilisation de la texture dans le contexte de l'évitement d'obstacles ou de la navigation [Betgé-Brezetz 96b]. Par exemple, Fernandez [Fernandez-Maloigne 95] présente une approche pour la détection rapide et automatique de routes, en utilisant une segmentation de l'image par une analyse de texture sur une architecture de réseau de neurones.

En outre, les effets de l'instabilité colorimétrique sur les capteurs sont plus prononcés à l'extérieur car l'information visuelle est fort dépendante de la géométrie (direction et intensité de la source lumineuse) et de la couleur (distribution de la puissance spectrale) de la lumière. Il est clair que la constance de la couleur à l'extérieur est très compliquée à obtenir surtout avec des conditions atmosphériques imprévisibles. Certaines approches [Celaya 02] préfèrent ainsi contourner ce problème en exploitant l'opposition ou le rapport de couleurs ; par exemple, pour atténuer les effets causés par des variations d'illumination, les rapports R/G et B/G ou l'espace *rgb* normalisé sont souvent utilisés.

Un des travaux les plus représentatifs de la navigation visuelle en milieu naturel au LAAS, est présenté par P. Lasserre [Lasserre 96] ; dans sa thèse, elle utilise le capteur caméra vidéo pour apporter des informations utiles à la localisation et à la navigation du robot dans son environnement. Elle a travaillé sur deux approches : l'obtention de l'information tridimensionnelle à partir d'un système stéréoscopique d'une part, l'identification de la nature des objets contenus dans la scène d'autre part.

Al Haddad, dans sa thèse [Haddad 98], considère la génération autonome de déplacements, à partir des informations fournies par une paire de caméras monochromes. Il a poursuivi les développements sur la stéréovision exploitée pour obtenir en ligne des données tridimensionnelles sur l'environnement, à partir desquelles une carte locale d'obstacles est déterminée : il a proposé un algorithme de stéréo-corrélation, technique qui est adaptée pour des scènes texturées. Il a

[10] Une région sur une image désigne une zone homogène en couleur et en texture, délimitée par un algorithme de segmentation

proposé aussi deux méthodes de génération de mouvements : l'une réactive, et l'autre « pseudo-planifiée ». Ces déplacements sont enchaînés afin d'atteindre un but situé à quelques dizaines de mètres, dans un environnement essentiellement plan.

R. Murrietta Cid a poursuivi les travaux sur l'interprétation des informations perçues à partir de la vision monoculaire couleur [Murrieta-Cid 98]. En effet, lorsque le robot évolue dans un environnement extérieur, la connaissance du terrain ou des objets situés dans cet environnement peut être améliorée en ajoutant des informations telles que la couleur ou la texture. Avec C. Parra, R. Murrietta a proposé une stratégie de navigation qui combine les informations 3D venant de la stéréo, et les informations 2D venant de la vision couleur, cela pour repérer et suivre pendant le mouvement du robot des amers de type rocher, buisson, ... La méthode développée [Murrieta-Cid 01] est fondée sur (1) la segmentation d'une image de profondeur en régions correspondant au terrain (surface uniforme) et aux entités qui en émergent (roches, chemin, végétation, ...), (2) sur l'extraction de caractéristiques sur ces régions, basées sur la couleur et la texture, (3) sur l'identification de la nature du terrain (terre, herbe) et des objets qui en émergent (rocher, arbres) et (4) sur le suivi des objets utiles pour le repérage du robot.

Notons que ces travaux n'ont pas été intégrés sur un robot, ce qui en limite singulièrement la portée.

En bref, du fait de la richesse des images, l'utilisation de la vision artificielle en robotique revêt une importance toute particulière car elle permet de fournir à la machine les capacités nécessaires pour réagir avec son environnement. Dans ce cadre, nous avons donc étudié des méthodologies d'extraction d'indices visuels dans les images permettant d'obtenir ensuite un guidage efficace et/ou une localisation précise d'un robot mobile par rapport à son univers.

1.5 Notre approche de navigation

Notre approche de navigation est caractérisée par l'utilisation d'une seule caméra pour construire un modéle $2D$ de l'environnement. Du modèle 2D, seront extraites les consignes qui permettront au robot de se guider. La méthode de modélisation 2D de l'environnement est fondée sur des techniques de segmentation et de classification, permettant l'extraction et la reconnaissance des principaux objets sur la scène. Ces techniques sont l'aboutissement de plusieurs travaux menés precedemment au sein du LAAS [Lasserre 96, Murrieta-Cid 98] ; notre contribution porte sur une plusieurs variantes vis-à-vis des méthodes proposées par R.Murrieta, sur une évaluation systématique des méthodes proposées, et sur le développement d'un module rapide de segmentation compatible avec l'architecture embarqué sur notre robot mobile Dala ; nous avons intégré ce module avec d'autres modules de vision, de planification ou de locomotion, et validé l'ensemble par plusieurs expérimentations.

Tout d'abord, les images acquises sont traitées numériquement pour les rendre adéquates à des tâches d'extraction d'indices visuels. L'image couleur est reconstruite à partir d'une image monochromatique brute (filtre de *Bayer*), en utilisant une méthode d'interpolation appelée démosaïquage (ou *demosaicking*). Les images couleur obtenues montrent généralement un déséquilibre chromatique caractérisé par la présence d'une couleur dominante (voile de couleur). L'opération chargée d'enlever cette couleur dominante est appelée la balance des blancs. Nous avons ainsi une image de bonne qualité, apte à être exploitée pour des fonctions visuelles nécessaires dans le processus de navigation.

L'image 1.2 illustre les traitements exécutés dans notre schéma de navigation. L'image couleur

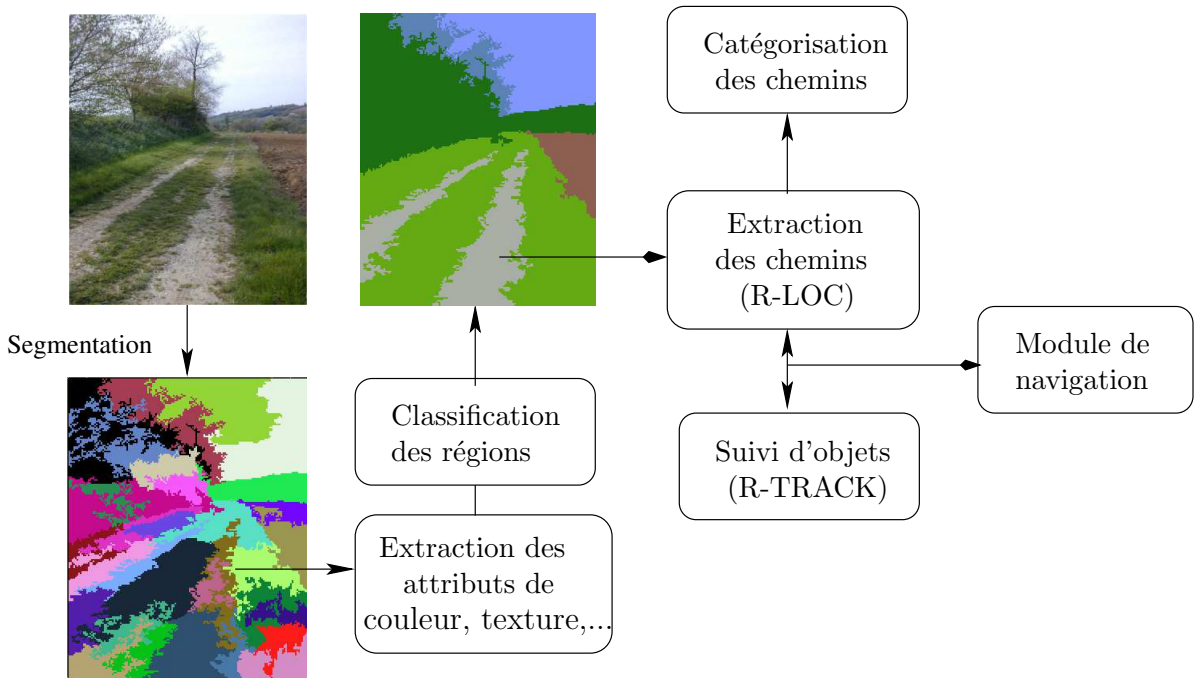


FIG. 1.2 – Notre méthode de navigation visuelle

est segmentée en utilisant une méthode hybride : seuillage (ou *c*lustering) et segmentation des régions (ou *r*egion growing).

Les régions de la scène sont alors représentées par un vecteur dans un espace $\mathbb{R}^{12}$, vecteur composé par des attributs de couleur et de texture et par l'information contextuelle. Ces vecteurs sont comparés à des échantillons préalablement appris pour chaque type de terrain ou d'objet à identifier ; nous utilisons une méthode de classification supervisée, soit la technique de *Support Vector Machines* (SVM), soit la technique classique des k-plus proches voisins (k-PPV). Pour augmenter le taux de reconnaissance et pour mieux conditionner les éléments formant la base de connaissances exploitée par les classifieurs, un pré-traitement basé sur l'analyse en composantes indépendantes (ACI) a été mis en oeuvre. La base de données est construite incrémentalement, c'est-à-dire que des nouvelles classes peuvent être facilement ajoutées au cours des expérimentations pour que le robot puisse s'adapter aux variations de l'environnement.

Une fois les régions identifiées, nous allons corriger les erreurs possibles de sur-segmentation, en fusionnant les régions voisines (connexes) ayant la même nature, donc appartenant à la même classe. Pour faire cela, un algorithme qui fait le balayage de la frontière entre deux régions, calcule le graphe d'adjacence des régions (RAG) permettant une fusion de régions rapide. C'est ainsi que la description $2D$ de la scène courante est obtenue par une opération complètement automatisée. Le résultat est un ensemble de régions et une image étiquetée (*cf.* figure 1.2), dans laquelle chaque région ou pixel porte le numéro de la classe identifiée ; une couleur particulière a été prédéfinie pour représenter visuellement chaque classe ; un numéro particulier est associé aux régions ou pixels non identifiés.

L'extraction des *chemins* ou des objets de n'importe quelle autre classe, revient à détecter les régions étiquetées par un numéro donné sur une image étiquetée. Il reste uniquement à extraire leur contour en utilisant un algorithme similaire à celui qui calcule le RAG. L'ensemble

de ce module visuel partant de l'acquisition d'une image couleur et finissant par l'extraction des contours des régions associées au chemin ou à un autre objet d'intérêt, est intégré sur le robot dans une module appelée *R-LOC*.

Les étapes suivantes dépendent de l'application souhaitée.

- Pour la construction d'un modèle topologique sur un réseau de chemins, nous proposons l'identification des chemins par leur forme, en utilisant à cet effet le descripteur « Shape Context ». Ceci nous permet d'assigner une catégorie aux chemins (ligne droite, virage, intersection...), ce qui est nécessaire pour la construction de graphes topologiques.
- Pour la navigation, nous identifions dans chaque image, la région chemin sur laquelle est le robot, et nous exploitons les frontières qui limitent cette région, pour estimer dans cette image, une trajectoire à suivre. Cette trajectoire dans l'image est convertie en une trajectoire sur le sol en utilisant l'hypothèse de terrain plat. Cette trajectoire est enfin exécutée par le robot ; à chaque itération, les portions de trajectoire non encore exécutées, sont fusionnées et filtrées avec les nouvelles, donnant de la stabilité au système.

Comme la fréquence d'exécution du module *R-LOC* est entre 1 et 2Hz selon la complexité de l'image, nous exploitons sur le robot, un module appelée *R-TRACK*, qui exécute une fonction de suivi visuel développée par A.Marin, fondée sur des contours actifs [Marin-Hernandez 04]. Ceci permet d'augmenter la cadence de notre système jusqu'à 10Hz. Les problèmes relatifs à la synchronisation entre ces deux taches ont été analysés et traités de manière simple.

1.6 Conclusion

Dans ce chapitre nous avons présenté d'abord l'état des travaux les plus réputés dans le domaine de la navigation visuelle. Ensuite, nous avons fait une distinction entre les travaux concernant les environnements naturels structurés et non structurés. L'évolution des travaux au LAAS dans ce domaine a été également décrite. Enfin, nous avons illustré l'approche proposée ainsi que nos contributions dans le domaine de la navigation visuelle de robots mobiles.

Nos travaux de recherche se placent donc dans la navigation visuelle des robots dans des environnements semi-structurés d'extérieur. En effet, l'utilisation des éléments comme les chemins de terre (produits par l'homme) pour la navigation implique un certain niveau de structuration, laquelle doit être enrichie avec de l'information de couleur, de texture et de forme, ainsi que des éléments contextuels. Dans ce cadre, nous étudions les méthodologies d'extraction d'indices visuels dans les images nous permettant d'obtenir ensuite une localisation précise et/ou un guidage efficace d'un robot mobile par rapport à son univers.

Dans le prochain chapitre, nous allons décrire la première étape de l'approche proposée pour l'interprétation des scène naturelles : la chaîne des pré-traitements à effectuer sur une image couleur acquise par une caméra munie d'un filtre *Bayer*.

If some day it becomes possible to recognize and to distinguish in an objective way the various effects of light by direct observation of the retina, people will perhaps recall with pitying the efforts of previous decades undertook to seek an understanding of the same phenomena by such lengthy detours

J. Von Kries père des modèles d'adaptation chromatique

Chapitre 2

Acquisition et reproduction d'images couleur

2.1 Introduction

Les caméras sont l'un des plus importants éléments dans la chaîne d'acquisition d'images, leurs propriétés et caractéristiques sont essentielles au bon déroulement du processus de perception en vision robotique. Spécifiquement, notre intérêt est focalisé sur les capteurs matriciels mono-CCD pour l'acquisition d'images numériques couleur [Mancuso 01].

Dans ce chapitre nous allons présenter les diverses problématiques liées à la reproduction de la couleur et les méthodes implémentées pour obtenir des images couleur de bonne qualité. Ces méthodes ont été intégrées dans le système embarqué sur le robot Dala pour traiter les images acquises par des caméras IEEE1394 de marque *Micropix C-1024*, mais nous les avons aussi appliquées à des images acquises par plusieurs appareils photos numériques.

Des fonctions comme le démosaïquage ou *demosaicking* [Kimmel 99], le calibrage automatique du point blanc (ou balance des blancs), l'adaptation chromatique, des transformations en ligne pour obtenir une invariance à l'illumination, la correction gamma [Tsin 01]... sont appliquées aux images couleur avant traitement. Connaître précisément ces fonctions et les optimiser sont des étapes fondamentales qui vont nous permettre d'exploiter des images couleur de qualité optimale, pour la reconnaissance d'objets et pour contrôler le déplacement d'un robot mobile en milieu naturel.

De plus, pendant l'exécution d'une de ces fonctions, la couleur de l'éclairage pour une scène peut changer : la couleur des surfaces présentes dans la scène va alors changer dans la même proportion. Ce décalage de couleur va être responsable de problèmes d'instabilité dans les descripteurs utilisés dans le système de vision. Évidemment, sans la stabilité colorimétrique de ces descripteurs, la plupart des applications impliquant la couleur (*e.g.*, systèmes de reconnaissance d'objets [Swain 91], photographie numérique [Jacobson 00], *etc.*) vont être négativement affectées par les moindres variations d'illumination [Funt 98].

On pourrait se demander pourquoi les performances du système visuel de l'homme ne sont pas affectées par de telles variations. Pour le savoir, la communauté scientifique [Brainard 97, Kraft 99] a essayé de mesurer expérimentalement la constance de couleur chez l'humain ; mais les mécanismes de la vision humaine menant à la constance des couleurs restent encore sans explication satisfaisante [Brainard 01, Rosenberg 01]. Pourtant, à l'intérieur de notre système visuel, chacun des photorécepteurs de la rétine est sensible à une seule gamme de longueur d'onde ; une seule composante couleur est échantillonnée à chaque position spatiale ; les caméras

mono-CDD utilisent ce même principe en fournissant uniquement une composante colorimétrique par pixel. Ce type d'échantillonage correspond à un multiplexage spatial de l'information couleur dans une image [Jacobson 00, Trémeau 04].

Dans les sections suivantes, nous décrirons tout d'abord les caméras que nous avons utilisées, toutes équipées de mosaïque *Bayer*. Nous décrirons et comparerons les méthodes existantes pour reconstituer une image couleur à partir de cette mosaïque. Puis, nous traiterons du calibrage chromatique, connu sous le nom *Balance des blancs*.

2.2 Acquisition des images couleur

L'acquisition d'une image couleur peut se faire à l'aide de plusieurs types de caméras. Dans les travaux qui ont précédé les notres dans notre groupe de recherche, R.Murrieta utilisait des caméras analogiques, soit des caméras mono-CCD délivrant une image vidéo au format PAL, soit des caméras tri-CCD qui donnaient directement les trois images au format RGB. Des cartes d'acquisition adaptées permettaient de récupérer sur le calculateur hôte, les trois plans image Rouge, Vert et Bleu. Rappelons que du fait du standard vidéo, la résolution d'une image analogique est limitée à 591 lignes ; le nombre de pixels échantillonés par ligne dépend de la taille de la matrice CCD et du numériseur. Avec les derniers numériseurs, R.Murrieta exploitait des images composées de 590 lignes de 768 pixels.

L'ère des caméras analogiques est révolue : nous n'avons exploité que des caméras CCD numériques. La limite sur la résolution des images est maintenant définie par la taille de la matrice CCD et par la vitesse de transmission de la liaison entre caméra et calculateur hôte. La technologie évolue très rapidement dans l'un et l'autre domaine, du fait des applications grand public : photo numérique (le moindre appareil a maintenant une matrice de 5Mega pixels), et Internet (pour accélérer les vitesses des liaisons entre modems ADSL et calculateur...). Dans notre cas,

- nos caméras sont connectées par bus série *Firewire* ou *IEEE 1394*, pour lesquelles les cartes interface sont déjà integrées sur les calculateurs en standard. Ce bus *Firewire* supporte des vitesses de transfert de l'ordre de 400Mb/s (soit 50Mo/s). L'un des avantages du FireWire est le branchement à chaud ; il constitue ainsi, l'interface idéal pour les équipements audio/vidéo numériques.
- les caméras numériques industrielles ont une résolution limitée par rapport aux appareils photo grand public, puisque typiquement, elles fournissent des images de 1M pixels. Nous n'exploitons aucun algorithme de compression d'images, ni aucun traitement interne à la caméra, cela pour avoir une totale maîtrise sur la qualité des images que nous allons exploiter.

Nous n'avons pas testé les caméras exploitant les deux autres protocoles qui existent à ce jour pour des caméras numériques :

- les caméras *USB*, souvent caméras bas coût de type *WebCam* ou très bas coût, comme les caméras construites par *STMicroelectronics* et intégrées dans les téléphones portables : dans notre groupe, les travaux sur la reconnaissance gestuelle ou le suivi de visages, exploitent de tels capteurs.
- les caméras *Camera Link* ou *LVDS*, qui au contraire, sont souvent exploitées pour des applications critiques (contrôle-commande, acquisition d'images haute résolution à plus de 30Hz ...) : dans notre groupe, le projet BODY SCAN (modélisation du corps humain) a fait usage de telles caméras.

Les caméras tri-CCD numériques n'étaient pas disponibles lorsque nous avons commencé nos travaux. Nous avons donc exploité uniquement des caméras mono-CCD, décrites dans la section suivante.

2.2.1 Caméras couleur fondées sur une mosaïque *Bayer*

Les capteurs CCD captent la lumière sur les petits photosites situés à leur surface. Les photosites sont organisés en rangées et colonnes, le plus souvent sur une matrice avec un passage vertical entre chacun d'entre eux de manière à ce que les charges électriques puissent être transférées de manière synchrone vers un registre. Pour des raisons techniques, ces détecteurs ne peuvent capter individuellement qu'une seule longueur d'onde à la fois en un photosite donné. La méthode la plus répandue pour obtenir des images couleur avec un capteur mono-CCD [Zomet 02] consiste à placer devant chaque cellule sensible, un filtre du type CFA[11], de telle sorte que chaque photosite du capteur CCD ne perçoit qu'une des trois composantes spectrales, généralement Rouge, Verte et Bleue (voir figure 2.1). Les pixels sont alors disposés selon un matriçage dit en quinconce.

Pour la mosaïque d'une matrice colorée, le filtre le plus utilisé est le 3-chromatique RGB, bien que d'autres soient aussi disponibles : le 3-couleurs complémentaires YeMaCy, le système à 4-couleurs où la quatrième couleur est le blanc ou une autre couleur d'une sensibilité spectrale décalée [Alleysson 02]. Bien que l'utilisation de plus de 3 composantes colorimétriques, dans la fabrication de capteurs CCD, semble donner plus d'informations spectrales sur la scène, la corrélation implicite entre les composantes de couleur réduit son utilité en pratique.

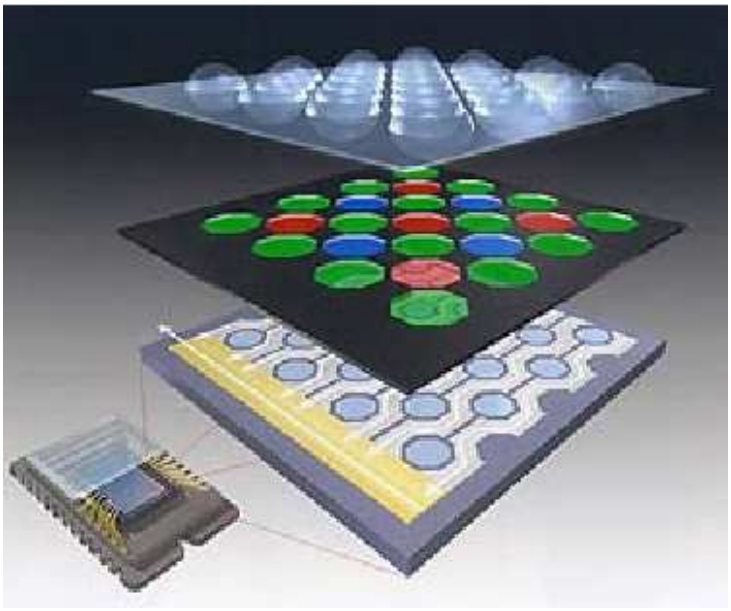
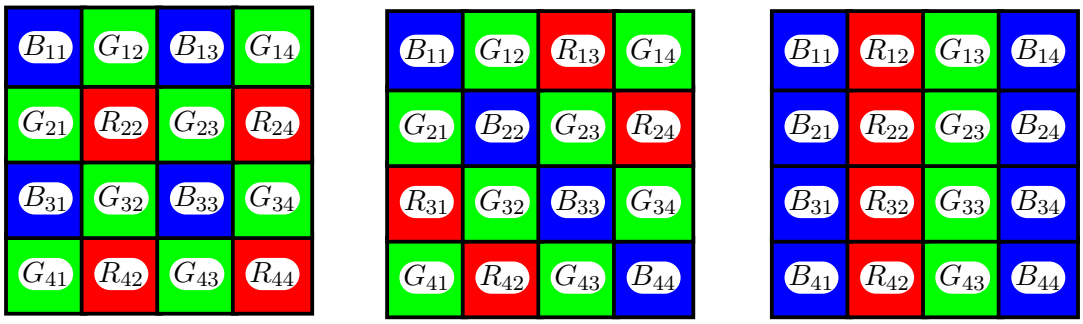


FIG. 2.1 – Caméra couleur avec mosaïque *Bayer*

On trouve sur la figure 2.2 deux exemples de filtres couleur, dont la mosaïque *Bayer* qui est celui le plus souvent utilisé dans la technologie des caméras numériques [Gunturk 04]. Ils permettent de calculer la valeur d'un pixel couleur par interpolation à partir d'un quadruplet de photosites, accordant le double d'importance à la composante verte (rouge et bleu sont

[11] Color Filter mosaic Array



(a) Mosaïque *Bayer* classique. (b) Mosaïque en diagonale. (c) Mosaïque par colonnes.

FIG. 2.2 – Les mosaïques de filtre couleur

échantillonnés avec une fréquence de 1/2 tant dans la direction horizontale qu'à la verticale tandis que le vert est échantillonné seulement à l'horizontale). Rappelons que le vert est la couleur pour laquelle le système visuel humain présente le maximum de sensibilité [Mancuso 01] ; en effet, de jour, la sensibilité spectrale maximale de l'oeil n'est pas uniforme, son maximum se situe dans le vert-jaune (550 nm). La conception des caméras mono-CCD s'est inspiré de ce fonctionnement biologique, ce qui explique le sur-échantillonnage du canal vert par rapport aux autres couleurs.

Ces types de capteurs présentent un avantage incontestable en réduisant le coût de fabrication d'une caméra traditionnelle tri-CCD pour laquelle trois matrices de photosites (une pour chaque bande) et une optique plus complexe sont utilisées. Par contre, les limitations des caméras basées sur mosaïque *Bayer* sont liées au fait qu'au moins trois cellules sont nécessaires pour obtenir par interpolation les couleurs manquantes, engendrant ainsi une perte de résolution [Kimmel 99]. De plus, les trois cellules utilisées pour compléter la couleur d'un pixel ne se localisent pas dans la même position ce qui peut provoquer des aberrations chromatiques (*e.g.*, production des fausses couleurs).

Le processus de reconstruction doit garantir un rendu d'images de haute qualité, en évitant en même temps les artefacts liés au processus d'acquisition.

2.2.2 Évolution des Caméras couleur

La technologie des capteurs n'arrête pas d'évoluer, et il en est de même de la technologie des caméras. En février 2002, la compagnie *Foveon* a introduit sur le marché la ligne de capteurs d'images *Foveon X3* [Merrill 03]. Ces capteurs sont les premiers au monde à capturer l'information couleur complète en chaque pixel dans une seule matrice photosensible (figure 2.3). Cette technologie permet une amélioration de la reproduction de la couleur et de la netteté des images. Ainsi, les erreurs provoquées par une opération d'interpolation (comme sur les caméras équipées de mosaïque *Bayer*) sont donc éliminées.

Le capteur X3 profite du fait que la lumière composée par des longueurs d'onde différentes est absorbée à différentes profondeurs dans le silicium. La couche des photorécepteurs bleus est positionnée près de la surface supérieure de l'élément photosensible, la verte au milieu et la rouge forme la dernière couche.

Parmi les évolutions technologiques récentes, signalons aussi, les progrès sur les matrices CMOS, connues pour avoir un rapport Signal sur Bruit plus mauvais que les matrices CCD, mais aussi pour permettre des capacités d'adressage aléatoire aux pixels et pour leur grande dynamique.

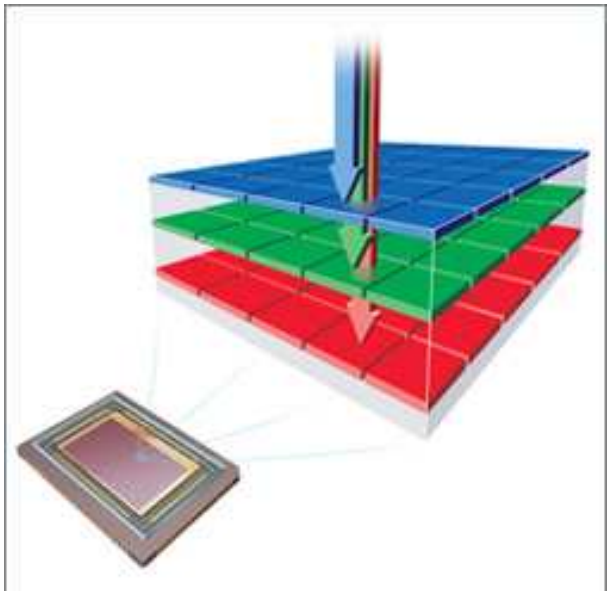


FIG. 2.3 – Nouvelle technologie des capteurs couleur *Foveon* X3.

Bien que ces nouvelles technologies soient théoriquement plus attractives et avantageuses, l'utilisation de caméras mono-CCD avec mosaïque *Bayer* reste toujours d'actualité, surtout dans le domaine de la photographie numérique de haute résolution. Toutes les images présentées dans ce mémoire ont donc été acquises par de tels capteurs, soit des caméras montées sur un robot, soit des appareils photo numérique. Dans ce dernier cas, l'image est directement stockée sur un fichier après avoir subi des traitements sur l'appareil (démosaïquage, filtrage, correction chromatique, compression). De même, il existe de nombreuses caméras faible coût *Firewire* qui font en interne plusieurs traitements : balance des blancs, génèration de l'image couleur et transmission au calculateur hôte au format 4x2x2 ou autre ; par exemple, dans notre équipe, les démonstrateurs *Scout* sont équipés de caméras Sony 500 exploitée au format 4x2x2.

Nous n'avons pas utilisé ces capteurs ; les caméras *Micropix C-1024* montées sur le robot DALA, ne transmettent que l'information brute, c'est-à-dire la luminance enregistrée en chaque photosite. Nous avons donc un contrôle total sur les pré-traitements qui doivent être faits dans le module d'acquisition avant toute analyse d'une telle image. La première étape consiste à reconstituer toute l'information chromatique en chaque photosite : dans le jargon des traiteurs d'images, on parle de démosaïquage, traduction de *demosaicking*.

2.3 Demosaïquage : reproduction des images couleur

Les algorithmes d'interpolation des matrices couleur transforment la sortie du capteur mono-CCD en une vraie image couleur (*i.e.*, RGB codé en 24 bits/pixel), en reconstruisant les trois composantes chromatiques sur chaque pixel (figure 2.4).

Plusieurs approches ont été proposées dans la littérature, mais malheureusement, la plupart d'entre elles sont brevetées [Cok 87, Freeman 88, Laroche 94, Hamilton Jr. 97, Adams Jr. 97, Crane 99] car implémentées dans plusieurs modèles d'appareils de photographie numérique : les informations sur ces approches sont donc introuvables. D'autres algorithmes pour réaliser cette interpolation, s'inspirent du système visuel humain. En effet la rétine de notre oeil, qui

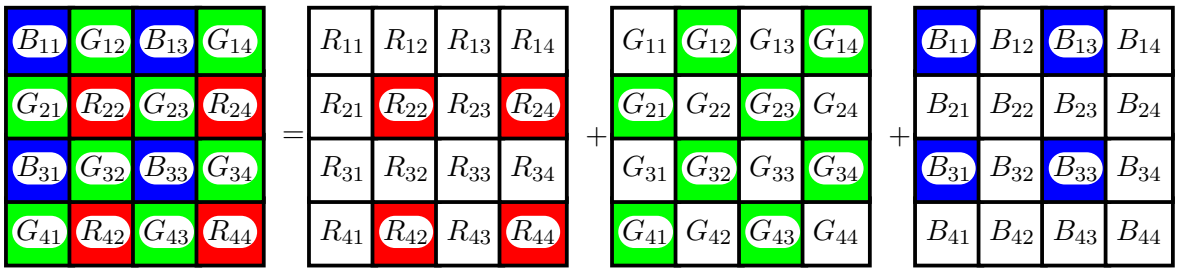


FIG. 2.4 – Demosaïquage : récupération de l'information couleur manquante

contient les récepteurs visuels, peut être comparée à une matrice de filtres couleurs, car chacun des photorécepteurs est sensible à une seule gamme de longueur d'onde [Ramanath 03]. Or, il a été montré par les expériences de psychologie que ces photorécepteurs ont une grande acuité spatiale en luminance et la couleur est codée par un processus d'opposition de couleur ; ces mécanismes sont exploités par certaines méthodes de démosaïquage [Alleysson 01].

La relation d'interpolation peut être exprimée par l'équation 2.1, où $C(x, y)$ représente le pixel interpolé, $c(x_k, y_l)$ est la valeur échantillonnée sur le pixel (x_k, y_l) et h représente le noyau d'interpolation appliqué sur une fenêtre MxN autour du pixel interpolé [Sakamoto 98].

$$c(x, y) = \sum_{k=1}^{M} \sum_{l=1}^{N} c(x_k, y_l) h(|x - x_k|) h(|y - y_k|) \tag{2.1}$$

Nous avons étudié le comportement de plusieurs algorithmes d'interpolation en fonction de critères comme la complexité algorithmique, le temps d'exécution, la qualité de la reconstruction visuelle (préservation des contours et limitation des fortes transitions de teinte) et l'impact sur la segmentation (ou classification) ; notre objectif est de sélectionner la méthode la plus adaptée pour notre problème de navigation.

Parmi les méthodes testées, nous ne détaillerons ici que les suivantes : une interpolation simple par le plus proche voisin, une interpolation bilinéaire, une interpolation non-linéaire par un filtrage médian, une méthode par teinte constante et deux méthodes adaptatives (fondées sur des flitres laplaciens) proposées pour préserver les contours.

2.3.1 Démosaïquage par « le plus proche voisin » (PPV)

Dans cet algorithme, la valeur des composantes R, G ou B pour le pixel interpolé, se calcule à partir de la valeur de l'un des pixels (de la même classe) le plus proche dans son voisinage, *i.e.*, leur implémentation implique une réplication des valeurs. Il est alors possible de choisir ce voisin, quelle que soit sa position : supérieure, inférieure, gauche ou droite. Le noyau d'interpolation [Sakamoto 98] pour cette méthode est régi par l'équation suivante :

$$h(x) = \begin{cases} 0 & 0 \leq x \leq 0,5\,, \\ 1 & x > 0,5\,. \end{cases} \tag{2.2}$$

Grâce à cette opération, appliquée sur tous les photosites de la matrice CCD, nous obtenons les trois plans couleur de la scène. En dépit de sa simplicité algorithmique, cette méthode a l'inconvénient de produire des images de qualité médiocre. Elle est donc déconseillée pour le traitement d'images.

2.3.2 Démosaïquage bilinéaire

Cette approche linéaire [Longère 02] utilise l'information des quatre pixels adjacents (de la même classe de photorécepteur) pour calculer le pixel manquant par moyennage. Le noyau de cette interpolation est formulé [Sakamoto 98] dans l'équation 2.3 et présenté graphiquement dans la figure 2.5,

$$h(x) = \begin{cases} 1 - x & 0 \leq x \leq 1\,, \\ 0 & 1 < x\,. \end{cases} \tag{2.3}$$

La composante verte est explicitement estimée par l'équation 2.4 sur les pixels R ou B, alors que les composantes rouge et bleue sont obtenues par les équations 2.5, selon la nature du pixel considéré.

$$h_G(x) = \left\{ \frac{1}{4} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad x \in \{x_R,\, x_B\}\,, \right. \tag{2.4}$$

$$h_R(x) = \begin{cases} \frac{1}{2} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} & x \in x_G \\ \frac{1}{4} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} & x \in x_B\,, \end{cases} \qquad h_B(x) = \begin{cases} \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} & x \in x_G \\ \frac{1}{4} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} & x \in x_R\,. \end{cases} \tag{2.5}$$

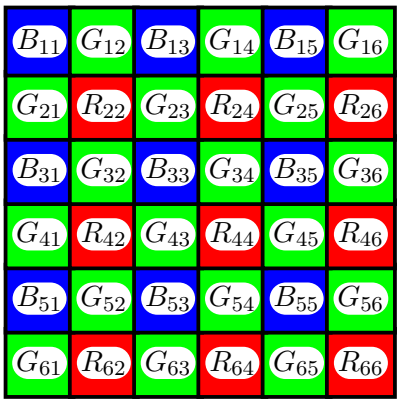


FIG. 2.5 – Mosaïque *Bayer* BGGR

À titre d'exemple,
- la composante verte G_{44} sur ce pixel équipé d'un filtre *Rouge* sur la figure 2.5 s'obtient par : $G_{44} = (G_{43} + G_{45} + G_{34} + G_{54})/4$.
- les composantes rouges (bleues) sur le pixel équipé d'un filtre *Vert* G_{34} sont estimées par : $B_{34} = (B_{33} + B_{35})/2$, $R_{34} = (R_{24} + R_{44})/2$.
- finalement, la composante rouge (bleue) sur un pixel équipé d'un filtre *Bleu* (*Rouge*) est estimée à partir de quatre pixels adjacents en diagonale, *e.g.*, $B_{44} = (B_{33} + B_{55} + B_{35} + B_{53})/4$ et $R_{33} = (R_{22} + R_{44} + R_{24} + R_{42})/4$.

Notons que ce processus est proche d'un filtre passe bas avec une bande passante limitée, ce qui génère un lissage colorimétrique des frontières en produisant des fausses franges colorées (crénelage, *zipper effect*) surtout sur les images texturées. Néanmoins, dans le cas du sous-échantillonnage, la prise en compte des quatre voisins rend la méthode moins sensible au bruit qu'avec l'approche PPV évoquée dans la section précédente.

2.3.3 Démosaïquage par filtrage médian (Freeman)

Ce type de démosaïquage [Freeman 88] consiste en deux étapes : la première est une interpolation linéaire, utilisée pour peupler chaque photosite sur les plans couleur ; la seconde étape est un filtrage médian appliqué sur les différences en couleur, $\boldsymbol{R}_{ij} - \boldsymbol{G}_{ij}$ et $\boldsymbol{B}_{ij} - \boldsymbol{G}_{ij}$. Les images issues du filtrage médian sont ensuite utilisées avec l'image mosaïquée d'origine pour reconstruire l'image de sortie [Ramanath 02].

Pour illustrer ceci, utilisons à nouveau la figure 2.5 pour présenter les équations qui permettent de calculer les composantes absentes sur les pixels $\boldsymbol{G}_{34}$ (calcul de $\boldsymbol{R}_{34}$ et $\boldsymbol{B}_{34}$), $\boldsymbol{R}_{44}$ (calcul de $\boldsymbol{G}_{44}$ et $\boldsymbol{B}_{44}$) et $\boldsymbol{B}_{33}$ (calcul de $\boldsymbol{G}_{33}$ et $\boldsymbol{R}_{33}$),

$$
\begin{array}{rlll}
\text{d'abord,} & \text{pour les composantes rouges :} & & \\
\boldsymbol{R}_{34} = \widehat{\boldsymbol{r}}_g + \boldsymbol{G}_{34} & \text{où,} & \widehat{\boldsymbol{r}}_g = med\left((\boldsymbol{R}_i - \boldsymbol{G}_i) |\, i \in \aleph_{G_{34}}\right) & \\
\boldsymbol{R}_{33} = \widehat{\boldsymbol{r}}_b + \boldsymbol{B}_{33} & \text{où,} & \widehat{\boldsymbol{r}}_b = med\left((\boldsymbol{R}_i - \boldsymbol{B}_i) |\, i \in \aleph_{B_{33}}\right) , & \\
\text{ensuite,} & \text{pour les composantes bleues :} & & \\
\boldsymbol{G}_{44} = \widehat{\boldsymbol{g}}_r - \boldsymbol{R}_{44} & \text{où,} & \widehat{\boldsymbol{g}}_r = med\left((\boldsymbol{R}_i - \boldsymbol{G}_i) |\, i \in \aleph_{R_{44}}\right) & (2.6) \\
\boldsymbol{G}_{33} = \widehat{\boldsymbol{g}}_b + \boldsymbol{B}_{33} & \text{où,} & \widehat{\boldsymbol{g}}_b = med\left((\boldsymbol{G}_i - \boldsymbol{B}_i) |\, i \in \aleph_{B_{33}}\right) , & \\
\text{et finalement,} & \text{pour les composantes vertes} & & \\
\boldsymbol{B}_{44} = \widehat{\boldsymbol{b}}_r + \boldsymbol{R}_{44} & \text{où,} & \widehat{\boldsymbol{b}}_r = med\left((\boldsymbol{R}_i - \boldsymbol{B}_i) |\, i \in \aleph_{R_{44}}\right) & \\
\boldsymbol{B}_{34} = \widehat{\boldsymbol{b}}_g + \boldsymbol{G}_{34} & \text{où,} & \widehat{\boldsymbol{b}}_g = med\left((\boldsymbol{G}_i - \boldsymbol{B}_i) |\, i \in \aleph_{G_{34}}\right) , &
\end{array}
$$

où *med* représente le filtrage médian sur le voisinage carré $\aleph$ autour du pixel à reconstruire.

Cette méthode présente une bonne qualité de préservation des contours et de reproduction de la couleur, mais sa complexité algorithmique est élevée à cause du filtrage médian dont le temps d'éxecution dépend de la taille de la fenêtre de voisinage (*i.e.*, 3×3, 5×5, 9×9, ...). Cette complexité est prohibitive pour nos applications en robotique. On constate que dans cet algorithme, la corrélation implicite des signaux de couleur s'utilise subtilement, ce qui est aussi exploité par plusieurs autres méthodes [Laroche 94, Adams Jr. 97].

2.3.4 Démosaïquage par teinte constante

Originalement proposé par David Cok, cet algorithme fut l'un des premiers à avoir été implémenté dans une caméra numérique commerciale, des variantes à ce système sont même encore employées. Dans le cas de l'interpolation bilinéaire, le problème principal est la présence d'artefacts colorés provenant du changement abrupt et peu naturel de la teinte sur la scène. Il est donc important d'éviter des fluctuations soudaines de la teinte à l'exception des pixels sur les contours [Cok 87].

Ici, les composantes rouge et bleue sont considérées comme l'information chromatique et la verte comme la luminance. Ainsi, la définition de teinte valide uniquement pour cette approche

est donnée par $\frac{R}{G}, \frac{B}{G}$. Évidemment, il faut faire attention aux singularités de cette formulation. Interpolant la valeur de la luminance pour le vert et se basant sur ce résultat pour dériver les teintes du bleu et du rouge, un changement graduel de la teinte est obtenu. Ceci permet une réduction des franges colorées (crénelage) [Gunturk 02].

Considérons une image à teinte constante. Les valeurs de luminance (G) et une composante chromatique (disons R) sur une position donnée par (R_{ij}, G_{ij}) sont liées à un autre échantillon sur une autre position (R_{kl}, G_{kl}) par la relation suivante :

$$\frac{R_{ij}}{R_{kl}} = \frac{G_{ij}}{G_{kl}} \quad \text{et} \quad \frac{B_{ij}}{B_{kl}} = \frac{G_{ij}}{G_{kl}} \tag{2.7}$$

dans un espace d'exposition logarithmique ou linéaire. Si R_{kl} est la valeur de chrominance inconnue, R_{ij} et G_{ij} sont les valeurs mesurées et G_{kl} est la luminance interpolée,

$$R_{kl} = G_{kl}\frac{R_{ij}}{G_{ij}} \tag{2.8}$$

Le démosaïquage commence par calculer d'abord toutes les composantes vertes pour les pixels *Rouge* ou *Bleu* de la matrice CCD, soit par la méthode bilinéaire soit par celle de Freeman. Cela nous permet de calculer la teinte et d'estimer les composantes R et B inconnues. Sur la figure 2.5, la composante pixel R_{33} (sur le pixel B_{33}) s'exprime par :

$$R_{33} = \frac{\boldsymbol{G}_{33}}{4}\left(\frac{R_{22}}{\boldsymbol{G}_{22}} + \frac{R_{24}}{\boldsymbol{G}_{24}} + \frac{R_{42}}{\boldsymbol{G}_{42}} + \frac{R_{44}}{\boldsymbol{G}_{44}}\right), \tag{2.9}$$

et la composante $B44$ (sur le pixel,R_{44}) par :

$$B_{44} = \frac{\boldsymbol{G}_{33}}{4}\left(\frac{B_{22}}{\boldsymbol{G}_{22}} + \frac{B_{35}}{\boldsymbol{G}_{35}} + \frac{B_{53}}{\boldsymbol{G}_{53}} + \frac{B_{55}}{\boldsymbol{G}_{55}}\right). \tag{2.10}$$

Afin de réduire la complexité de calcul, l'utilisation de logarithmes nous permet de changer les opérations de division en soustraction dans tout le traitement de l'image. En pratique, il est conseillé d'utiliser initialement la méthode de Freeman pour réduire les artefacts lors de l'estimation du vert. De ce fait, la complexité algorithmique de cette méthode est élevée, ce qui est prohibitif pour nos applications en Robotique.

Jusqu'ici nous n'avons décrit que des approches non adaptatives, donc qui font le même traitement sur toute l'image, ce qui a des effets négatifs sur les contours. Les sections suivantes présentent donc deux approches adaptatives : la dernière est celle que nous avons intégrée sur notre robot.

2.3.5 Démosaïquage par détection de gradients

Les méthodes décrites ci-dessus font une interpolation basée sur le moyennage des pixels autour d'un voisinage, ce qui provoque des effets de crénelage. Des méthodes adaptatives plus robustes exploitent l'information spatiale sur le voisinage en définissant un prédicteur local.

Laroche et Prescott [Laroche 94] ont proposé une approche basée sur la définition d'un gradient local. Comme pour l'approche de teinte constante, cette méthode repose sur deux étapes : la première consiste à obtenir la composante verte sur tous les photosites ; ces composantes seront ensuite utilisées lors de la deuxième étape, pour calculer les valeurs de chrominance (rouge et bleu). On définit ainsi un gradient qui dépend exclusivement de la luminance ; ce choix est justifié sur le fait que l'oeil humain est plus sensible aux changements d'intensité.

Utilisons encore la figure 2.5 pour illustrer la démarche de l'algorithme, d'abord pour l'obtention des valeurs de luminance. Afin d'obtenir la composante $\boldsymbol{G}_{44}$ (sur un pixel rouge), il a fallu définir des estimateurs (dérivées secondes) qui permettent la détection des contours (discontinuités) sur le canal vert, *i.e.*, $\alpha = |(\boldsymbol{R}_{42} + \boldsymbol{R}_{46})/2 - \boldsymbol{R}_{44}|$ et $\beta = |(\boldsymbol{R}_{24} + \boldsymbol{R}_{64})/2 - \boldsymbol{R}_{44}|$. Ceux-ci permettent en même temps d'établir s'il s'agit d'un contour vertical ou horizontal. Dans notre exemple, nous avons les relations suivantes pour l'obtention de la composante verte inconnue :

$$\mathbf{G}_{44} = \begin{cases} \dfrac{\boldsymbol{G}_{43} + \boldsymbol{G}_{45}}{2} & \text{si } \alpha < \beta \\ \dfrac{\boldsymbol{G}_{34} + \boldsymbol{G}_{54}}{2} & \text{si } \alpha > \beta \\ \dfrac{\boldsymbol{G}_{43} + \boldsymbol{G}_{45} + \boldsymbol{G}_{34} + \boldsymbol{G}_{54}}{4} & \text{si } \alpha = \beta\,. \end{cases} \tag{2.11}$$

De même, pour l'estimation d'une composante verte $\boldsymbol{G}_{33}$ (cette fois sur un photosite bleu), nous utilisons les relations spatiales du voisinage formé par des photosites bleus, $\alpha = |(\boldsymbol{B}_{31} + \boldsymbol{B}_{35})/2 - \boldsymbol{B}_{33}|$ et $\beta = |(\boldsymbol{B}_{13} + \boldsymbol{B}_{53})/2 - \boldsymbol{B}_{33}|$

$$\mathbf{G}_{33} = \begin{cases} \dfrac{\boldsymbol{G}_{32} + \boldsymbol{G}_{34}}{2} & \text{si } \alpha < \beta \\ \dfrac{\boldsymbol{G}_{23} + \boldsymbol{G}_{43}}{2} & \text{si } \alpha > \beta \\ \dfrac{\boldsymbol{G}_{32} + \boldsymbol{G}_{34} + \boldsymbol{G}_{23} + \boldsymbol{G}_{43}}{4} & \text{si } \alpha = \beta\,. \end{cases} \tag{2.12}$$

Grâce aux valeurs de luminance préalablement calculées, ce procédé exploite une interpolation sur les valeurs de chrominance en utilisant les différences de couleur $(R-B)$ et la luminance $\mathbf{G}$ préalablement calculée. L'équation 2.13 présente l'estimation de certaines composantes bleues inconnues ; les composantes rouges inconnues peuvent être calculées de la même manière.

$$\begin{aligned} \boldsymbol{B}_{34} &= \frac{(\boldsymbol{B}_{33} - \mathbf{G}_{33}) + (\boldsymbol{B}_{35} - \mathbf{G}_{35})}{2} + \boldsymbol{G}_{34}\,, \\ \boldsymbol{B}_{43} &= \frac{(\boldsymbol{B}_{33} - \mathbf{G}_{33}) + (\boldsymbol{B}_{53} - \mathbf{G}_{53})}{2} + \boldsymbol{G}_{43}\,, \\ \boldsymbol{B}_{44} &= \frac{(\boldsymbol{B}_{33} - \mathbf{G}_{33}) + (\boldsymbol{B}_{35} - \mathbf{G}_{35}) + (\boldsymbol{B}_{55} - \mathbf{G}_{55}) + (\boldsymbol{B}_{53} - \mathbf{G}_{53})}{4} + \boldsymbol{G}_{44}\,, \end{aligned} \tag{2.13}$$

Ainsi, cette technique qui exploite l'interpolation par différence de couleurs et le fait d'ajouter la composante verte, fournit des améliorations visuelles significatives car l'information couleur est estimée tout en respectant les contours.

2.3.6 Interpolation adaptative par laplacien

La dernière approche que nous décrirons, est fondée sur des modifications de la méthode décrite dans la section précédente ; elle a été proposée par [Hamilton Jr. 97]. L'originalité de cet algorithme consiste à combiner deux classifieurs qui exploitent des dérivées premières et secondes, calculées autour d'un voisinage. L'estimation des pixels inconnus dépend encore de l'orientation du contour, identifiée par ces descripteurs.

Cette méthode est encore organisée en deux étapes : la première étape consiste toujours à estimer la composante de luminance G et la deuxième à estimer les composantes chromatiques R et B. Pour illustrer le déroulement de l'algorithme, nous utilisons à nouveau la figure 2.5 et

le voisinage de pixels autour de $\boldsymbol{R}_{44}$. Les attributs horizontal α et vertical β sont définis de la façon suivante :

$$\begin{aligned} \alpha &= |\mu_x| + |(\boldsymbol{G}_{43} - \boldsymbol{G}_{45})|, \quad \mu_x = \boldsymbol{R}_{44} - \boldsymbol{R}_{42} + \boldsymbol{R}_{44} - \boldsymbol{R}_{46} \\ \beta &= |\mu_y| + |(\boldsymbol{G}_{34} - \boldsymbol{G}_{54})|, \quad \mu_y = \boldsymbol{R}_{44} - \boldsymbol{R}_{24} + \boldsymbol{R}_{44} - \boldsymbol{R}_{64}. \end{aligned} \tag{2.14}$$

où les termes μ_x et μ_y représentent les dérivées secondes autour d'un voisinage des pixels du même type (ici, rouge) et les termes complémentaires expriment les dérivées premières dans les orientations horizontale et verticale, en utilisant les informations des photosites G voisins du pixel traité (ici $\boldsymbol{R}_{44}$).

Dans la première étape, pour l'obtention de la composante de luminance $\boldsymbol{G}_{44}$ sur ce pixel chromatique rouge (ce serait pareil pour le pixel bleu $\boldsymbol{B}_{33}$), l'algorithme exploite les deux attributs α et β pour classifier le type du contour en ce pixel,

$$\mathbf{G}_{44} = \begin{cases} \dfrac{\boldsymbol{G}_{43} + \boldsymbol{G}_{45}}{2} + \dfrac{\mu_x}{4} & \text{si } \alpha < \beta \\ \dfrac{\boldsymbol{G}_{34} + \boldsymbol{G}_{54}}{2} + \dfrac{\mu_y}{4} & \text{si } \alpha > \beta \\ \dfrac{\boldsymbol{G}_{43} + \boldsymbol{G}_{45} + \boldsymbol{G}_{34} + \boldsymbol{G}_{54}}{4} + \dfrac{\mu_x + \mu_y}{8} & \text{si } \alpha = \beta\,. \end{cases} \tag{2.15}$$

Ces derniers estimateurs sont composés par le moyennage des pixels verts, de manière similaire à la méthode de la section précédente, cette fois complémenté par les valeurs pondérées de leurs dérivées secondes.

Lors de la deuxième étape, l'algorithme calcule les composantes chromatiques (rouge/bleue) manquantes. D'abord, nous calculons les composantes rouge et bleue sur un photosite vert (par exemple $\boldsymbol{G}_{34}$), en utilisant aussi les dérivées premières localement calculées,

$$\begin{aligned} \boldsymbol{R}_{34} &= \frac{\boldsymbol{R}_{24} + \boldsymbol{R}_{44}}{2} + \frac{\boldsymbol{G}_{34} - \boldsymbol{G}_{24} + \boldsymbol{G}_{34} - \boldsymbol{G}_{44}}{4} \\ \boldsymbol{B}_{34} &= \frac{\boldsymbol{B}_{33} + \boldsymbol{B}_{35}}{2} + \frac{\boldsymbol{G}_{34} - \boldsymbol{G}_{33} + \boldsymbol{G}_{34} - \boldsymbol{G}_{35}}{4} \end{aligned} \tag{2.16}$$

Dans l'interpolation des pixels chromatiques, la dernière situation possible consiste à estimer une composante bleue (resp. rouge) sur un photosite rouge (resp. bleu). Pour ce faire, il est à nouveau indispensable de définir un autre classifieur (en diagonale) qui nous permet de conserver les contours. Par exemple, afin d'estimer la composante $\boldsymbol{R}_{33}$ sur le photosite bleu $\boldsymbol{B}_{33}$, il faut au préalable obtenir les attributs Δx et Δy qui permettront de classifier l'orientation du contour chromatique en ce pixel,

$$\begin{aligned} \Delta x &= |\boldsymbol{R}_{22} - \boldsymbol{R}_{44}| + |\boldsymbol{\delta}_x|\,, \quad \boldsymbol{\delta}_x = \boldsymbol{G}_{33} - \boldsymbol{G}_{22} + \boldsymbol{G}_{33} - \boldsymbol{G}_{44}\,, \\ \Delta y &= |\boldsymbol{R}_{24} + \boldsymbol{R}_{42}| + |\boldsymbol{\delta}_y|\,, \quad \boldsymbol{\delta}_y = \boldsymbol{G}_{33} - \boldsymbol{G}_{24} + \boldsymbol{G}_{33} - \boldsymbol{G}_{42}\,, \end{aligned} \tag{2.17}$$

L'algorithme proprement dit pour obtenir la composante chromatique manquante (ici $\boldsymbol{R}_{33}$) suit les règles suivantes :

$$\boldsymbol{R}_{33} = \begin{cases} \dfrac{\boldsymbol{R}_{22} + \boldsymbol{R}_{44}}{2} + \dfrac{\boldsymbol{\delta}_x}{2} & \text{si } \Delta x < \Delta y \\ \dfrac{\boldsymbol{R}_{24} + \boldsymbol{R}_{42}}{2} + \dfrac{\boldsymbol{\delta}_y}{2} & \text{si } \Delta x > \Delta y \\ \dfrac{\boldsymbol{R}_{22} + \boldsymbol{R}_{44} + \boldsymbol{R}_{24} + \boldsymbol{R}_{42}}{4} + \dfrac{\delta_x + \delta_y}{4} & \text{si } \Delta x = \Delta y\,. \end{cases} \tag{2.18}$$

Ces derniers estimateurs sont calculés par la moyenne des pixels rouges (ou bleus) réajustés par la valeur pondérée de la dérivée seconde. Il est évident que cette méthode suit mieux les possibles discontinuités dans l'image d'origine à cause des critères utilisés lors de l'interpolation.

2.3.7 Comparaison entre les techniques de démosaïquage

Nous allons synthétiser les caractéristiques les plus distinctives des méthodes que nous avons testées. Les propriétés qui nous intéressent ici, sont la complexité apparente des algorithmes et la qualité de l'image obtenue (préservation de contours), *etc.* , vis à vis des besoins d'une application robotique pour laquelle les ressources sont limitées.

Premièrement, la méthode d'interpolation par le plus proche voisin ou « réplicateur » se caractérise par des calculs rapides, les niveaux sont conservés mais le résultat visuel est assez critiquable car la notion de contours est négligée, ce qui la rend inadaptée pour les images très texturées acquises en milieu naturel.

Le démosaïquage bilinéaire est une bonne option pour une reproduction rapide des images couleur ; au moins donne-t-il des résultats qui semblent réalistes à l'oeil nu. Son principal inconvénient ressort au moment de calculer les pixels sur les contours car ils sont moyennés et produisent des problèmes de crénelage. Dans le cas des images texturées, l'image de sortie devient un peu floue du fait de la perte des hautes fréquences.

La méthode de Freeman produit des résultats satisfaisants au niveau colorimétrique et des contours ; elle est robuste afin de corriger le bruit dit de *speckle*. Mais on constate une importante complexité algorithmique qui est due principalement au filtrage médian. Globalement, elle donne des résultats à peine meilleurs que la méthode de Laroche et Prescott [Laroche 94].

L'algorithme d'interpolation par teinte constante produit des résultats bien meilleurs que l'interpolation bilinéaire surtout si l'image de luminosité (verte) se calcule avec une méthode adaptative ou par un filtrage médian ; en effet, pour éviter de calculer des mauvais attributs, il est déconseillé de l'implémenter avec un filtrage bilinéaire. D'après nos expériences, l'image d'intensité obtenue est meilleure avec la méthode de Laroche ; ensuite les composantes chromatiques sont récupérées comme il est originalement proposé en section *cf.* § 2.3.4. Cependant, cette méthode nécessite une vérification et une correction globale des problèmes de débordement (*overflow*) lors du calcul des composantes chromatiques, ralentissant de fait beaucoup la reconstruction de l'image.

Nous savons que le système visuel humain est très sensible aux contours. Les approches qui essayent activement de les préserver, se regroupent dans l'ensemble des méthodes adaptatives. Deux telles méthodes ont été implémentées :

- l'algorithme de Laroche qui utilisent un détecteur de contour basé sur des dérivées secondes (*edge sensing*), ce qui lui permet de bien conserver les bords. Sa complexité algorithmique est raisonnable par rapport à la bonne qualité de reproduction d'images.
- la méthode de Hamilton fondée aussi sur un détecteur de gradient « intelligent » très similaire au gradient utilisé par Laroche. Mais, ici les dérivées premières et secondes sont exploitées pour caractériser les points de contour, ce qui améliore la sensibilité pour la détection des contours. Par ailleurs c'est la seule méthode étudiée qui utilise un gradient en diagonale ce qui a permis la meilleure reconstitution d'images couleur 24 bits parmi les méthodes implémentées.

Nous avons identifié plusieurs autres techniques qui semblent très intéressantes à implémenter, mais malheureusement leur complexité algorithmique indiquée par leurs auteurs, les rendent, aujourd'hui, inadaptées aux applications de navigation visuelle [Alleysson 01]. Ces nouvelles approches sont détaillées dans [Gunturk 04, Gunturk 02, Alleysson 02, Ramanath 03], une inter-

polation dite « optimale » est proposée par Muresan [Muresan 02], des redéfinitions de gradients pour la détection de l'orientation du contour sont présentées par Adams [Adams Jr. 97] et Chang [Chang 99].

Pour l'instant, c'est la méthode de Hamilton qui est intégrée dans notre chaîne de traitement.

2.4 Calibration chromatique d'images numériques

2.4.1 La couleur dans la reconnaissance des objets

Dans la plupart des applications, le dernier objectif d'un système de vision consiste à obtenir l'interprétation la plus complète du contenu des images. Un tel processus implique l'utilisation de techniques robustes pour l'estimation des attributs discriminants, pour l'étude des relations géométriques, pour l'analyse des dépendances entre les objets sur la scène [Zhong 00] ... La couleur se révèle être un attribut discriminant dans l'indexation [Rubner 98] et l'identification d'objets. Il est fort souhaitable que ce puissant attribut[12] soit le plus stable possible afin de rendre possible l'implémentation d'un système robuste ayant un large spectre d'applications.

En effet, la reconnaissance d'objets peut devenir difficile s'il n'existe pas une relation simple entre les propriétés de l'objet et son image rétinienne. La position de l'objet, son orientation et la manière par laquelle il est illuminé sont des facteurs qui affectent également l'image [Geusebroek 01]. De plus, cette relation peut être sous déterminée : plusieurs configurations physiques peuvent nous amener à la même image rétinienne [Sharma 97]. Les propriétés de l'image qui requièrent typiquement un certain type de correction sont la couleur, le contraste et la netteté (sharpness).

2.4.1.1 Physique de la couleur

Dans le cas spécifique d'images couleur, la distribution de la puissance spectrale de la lumière réfléchie par l'objet dépend non seulement de la réflectivité intrinsèque de sa surface mais également des facteurs extrinsèques, tels que le type d'illumination et la réflectivité (lumière réfléchie et altérée) des autres objets [Brainard 97].

La lumière incidente sur une surface est caractérisée par sa distribution de puissance spectrale $E(\lambda)$. Un petit élément de la surface de l'objet réfléchit une fraction de l'illumination incidente dans l'oeil. La fonction de réflectivité $S(\lambda)$ établit cette fraction comme une fonction de la longueur d'onde de la lumière incidente. Le spectre de la lumière qui atteint l'oeil est connu comme le signal couleur, représenté mathématiquement par

$$C(\lambda) = E(\lambda)S(\lambda)\,. \tag{2.19}$$

L'information implicite dans $C(\lambda)$ est codée par l'oeil, en utilisant trois classes de cônes photorécepteurs *L, M* et *S*. La figure 2.6 illustre la réponse spectrale donnée par ces cônes, sensibles aux longueurs d'onde lages (L), medium (M, noté aussi I car assimilé à l'intensité lumineuse) et courte (S) : on constate que les sensibilités des cônes *L, M* et *S* ne sont pas exactement centrées sur les trois couleurs fondamentales *R, G* et *B*.

[12] l'oeil humain peut discerner des milliers de tonalités en couleur, mais seulement deux douzaines de tons de gris [Kraft 99].

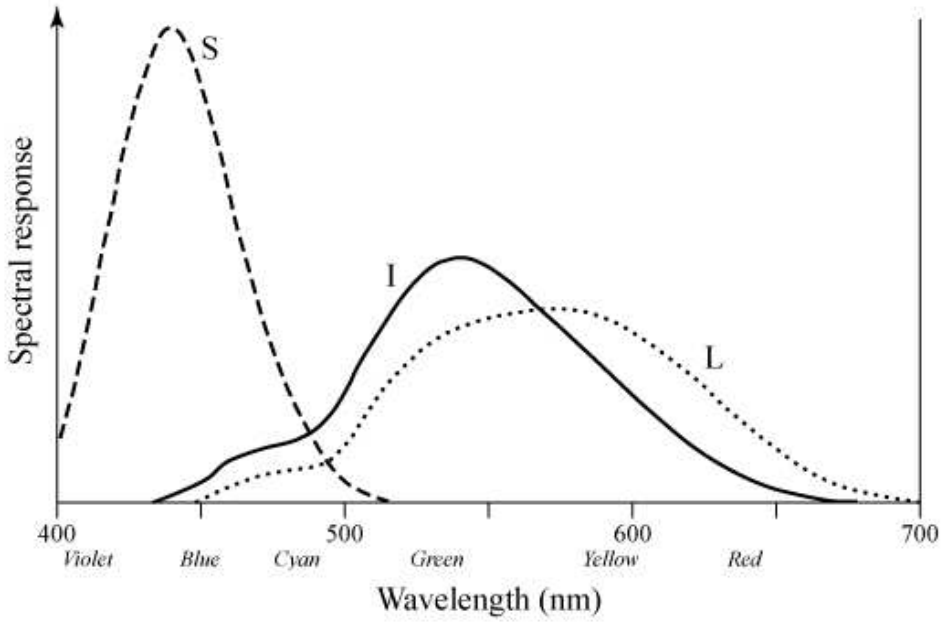


FIG. 2.6 – Réponse spectrale des trois cônes à la lumière

2.4.1.2 Constance de couleur

D'une façon ou d'une autre, les systèmes de vision doivent stabiliser l'apparence de couleur des objets contre les changement d'illumination, cet effet perceptuel est connu comme la constance de couleur ou *color constancy*. Puisque l'illumination est le facteur le plus saillant qui affecte le signal de couleur, il est naturel d'essayer de comprendre comment les changements d'illumination affectent l'aspect de la couleur.

Dans une expérience ordinaire sur la constance de couleur, l'illumination est considérée comme la variable indépendante : la variable mesurée, dépendante de l'illumination, est une mesure quelconque de l'apparence de couleur [Lucassen 97, Brainard 01]. La constance de couleur est un problème sous déterminé et par conséquence il n'est pas possible de le résoudre de manière générale. Plusieurs stratégies sont disponibles dans la littérature essayant d'obtenir des solutions approximatives [Funt 98, Brainard 97]. Ces expériences emploient des configurations des stimulus et tâches psychophysiques différentes ; elles montrent globalement que la vision de l'être humain manifeste un considérable degré de constance vers la couleur, mais les mécanismes impliqués dans la constance de couleur sont encore méconnus.

En bref, dans des applications de vision par ordinateur, en particulier la reconnaissance d'objets, les descripteurs en couleur doivent être indépendants aux changements d'éclairage. Un premier problème à résoudre est la correction d'un voile chromatique : en effet, avec les caméras numériques que nous exploitons, un voile constituée d'une couleur non souhaitée, est souvent superposé aux images acquises sur des scènes dont l'illumination n'est pas maîtrisée.

2.4.1.3 Notre problème

Alors que la perception de la couleur par le système visuel humain est peu affectée par le type d'éclairage (naturel, incandescent, fluorescent, halogène, ...), la vision artificielle (basée sur la technologie CCD) est très influencée par les conditions d'illumination [Tsin 01].

En effet, nous avons détecté une forte couleur dominante (un voile) sur les images reconstruites par démosaïquage après l'acquisition. Celles-ci devenaient jaunes sur un éclairage artificiel, tandis que sous un éclairage naturel elles montraient un voile rose (ou violet). Ce déséquilibre chromatique était si prononcé que même notre procédure de segmentation échouait au moment de dégager les objets principaux dans la scène (voir figure 2.7(b)). Il est donc évident que les images acquises et démosaïquées ne sont pas encore exploitables et un post-traitement correcteur de la couleur est toujours nécessaire.

(a) Image brute : mosaïque *Bayer*

(b) Image démosaïquée et voilée

(c) Image corrigée chromatiquement

FIG. 2.7 – Séquence de la reproduction des images numériques

Enlever le voile coloré[13] est une opération connue sous le nom *B*alance des Blancs. Théoriquement, la complexité inhérente à la balance des blancs consiste à détecter les composantes chromatiques qui provoquent le déséquilibre chromatique.

Dans les sections suivantes, nous détaillons les techniques utilisées pour estimer les facteurs qui amènent à la correction de la couleur, les modèles d'adaptation chromatique basés sur l'hy-

[13] *Color casting*, couleur dominante superposée due aux conditions d'illumination ou aux caractéristiques du capteur

pothèse de Von Kries et l'effet de la correction gamma[14] qui nous a servi de source d'inspiration pour améliorer l'aspect de nos images.

2.4.2 La balance des blancs

L'équilibrage de couleur s'avère utile à la suppression d'une polarisation globale des couleurs (tonalité dominante) d'une image, ce qui est aussi connue comme le réglage de la balance des couleur [Gasparini 03]. L'apparition des tonalités (voiles) dominantes peut résulter de causes différentes : la manière dont la scène originale a été illuminée, le film et les filtres que nous utilisons, les variations du film traitant et l'impression, les propriétés intrinsèques des capteurs CCD ou du procédé de balayage.

Puisqu'il est difficile ou impossible de commander tous les facteurs qui peuvent créer un déséquilibre de couleur, il est habituellement plus facile de corriger le problème à la fin du processus. Deux méthodes permettent d'adoucir une couleur : la première consiste à adoucir directement cette couleur particulière, et la deuxième à renforcer le ton des autres couleurs. En fait, ces deux méthodes doivent être combinées. En effet, si nous nous contentons d'adoucir une couleur primaire, telle que le rouge, ceci peut dégrader la qualité globale de l'image. Pour corriger ce phénomène, on essaye également de renforcer le vert et le bleu. Dans ce cas, les tons rouges seront relativement adoucis, affectant donc le rouge, mais la balance des couleurs de l'ensemble de l'image sera conservée.

Evidemment, la détection et la mesure automatique de cette tonalité dominante ont des complexités algorithmiques importantes ; parfois la couleur des images est perceptuellement compliquée à équilibrer même par un oeil expert. Théoriquement, l'effet de plusieurs illuminants devrait être annulé seulement dans le domaine spectral ce qui s'avère peu adapté dans la majorité des applications courantes. En conséquence, la plupart des algorithmes pratiques sont développés en utilisant une approximation dans les espaces de couleur 3D. La solution que nous avons adoptée utilise des grandeurs statistiques issues de l'image originale [Gasparini 04, Tsin 01] et l'hypothèse de Von Kries [Kries 93].

2.4.2.1 Hypothèse de J. Von Kries

Chez l'être humain, le système visuel qui contrôle l'adaptation des photorécepteurs au type d'illumination courante pour préserver l'apparence d'un objet a été expliqué par Johannes Von Kries au début du vingtième siècle. Il considérait que le mécanisme d'adaptation chromatique pouvait s'expliquer uniquement par le contrôle individuel du gain sur les cônes des photorécepteurs (L, M, S) de la rétine et que la grandeur du gain était une fonction directe de l'illumination courante [Kries 93, Fairchild 98].

Mathématiquement, le modèle de Von Kries est représenté par l'équation 2.20, où $\boldsymbol{K}_L, \boldsymbol{K}_M$ et $\boldsymbol{K}_S$ sont les paramètre d'adaptation (gains) qui permettront aux cônes L_1, M_1 et S_1 (l'excitation des couleurs sur une illumination de départ) d'adapter leur sensibilité en fonction d'une source d'éclairage quelconque.

$$\begin{pmatrix} L_2 \\ M_2 \\ S_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{K}_L & 0 & 0 \\ 0 & \boldsymbol{K}_M & 0 \\ 0 & 0 & \boldsymbol{K}_S \end{pmatrix} \begin{pmatrix} L_1 \\ M_1 \\ S_1 \end{pmatrix} \tag{2.20}$$

Bien qu'actuellement, ce modèle soit considéré comme incomplet [Fairchild 96, Funt 03], il sert et a servi d'inspiration à la plupart des modèles d'adaptation chromatique les plus sophis-

[14] critère définissant le caractère non-linéaire de l'intensité lumineuse d'un élément

tiqués utilisés en vision (*Chromatic Adaptation Models* ou CAT's) ; rappelons que l'adaptation peut être considérée comme une mise au point du système visuel humain pour optimiser la réponse visuelle vers un point particulier d'observation.

En fait, on ne peut pas nier que la relation de Von Kries est tout de même un bon prédicteur de base pour de nombreux cas, mais il manque au modèle de Von Kries, un processus d'adaptation au *contraste* qui prendrait en compte aussi l'interdépendance (ou corrélation) parmi les photorécepteurs et les possibles non linéarités lors du traitement visuel.

Notons que ces méthodes vont « corriger » chromatiquement l'image de la scène mais aucune d'entre elles n'est capable d'estimer en général directement les grandeurs du gain qui ramèneraient l'impression visuelle de la scène à l'équilibre. Dans les sections 2.4.3.1 à 2.4.3.4 ci-après, nous décrivons les approches courantes pour détecter le déséquilibre chromatique sur une scène, nous permettant d'estimer $\boldsymbol{K}_L, \boldsymbol{K}_M$ et $\boldsymbol{K}_S$.

2.4.3 Détection de la couleur dominante

2.4.3.1 Hypothèse du *monde gris*

Quand la luminance moyenne de la scène change, notre système visuel exécute divers mécanismes pour s'adapter au nouvel état. Ceci nous permet d'aboutir à une perception comparable de luminance en dépit de ce changement. Dans la photographie ce phénomène s'obtient par modification de l'ouverture de l'objectif et réglage du temps de l'obturateur, afin d'acquérir l'image avec les niveaux de gris adaptés à la dynamique disponible [Rizzi 03].

Lorsque, dans le domaine du traitement d'images, un déséquilibre chromatique se produit, celui-ci peut être causé par le manque d'un mécanisme d'adaptation chromatique. Quand ceci se produit de façon indépendante entre chacun des trois plans couleur, il est possible d'éliminer globalement cette dominante chromatique parasite en modifiant le gain de chacun des plans. En effet, si une *quantité suffisante de couleurs* est présente sur une image, la couleur moyenne sur toute celle-ci devra être proche du gris. En fonction de la valeur de cette moyenne, il sera possible d'estimer les gains qui corrigeront chromatiquement notre image. Nous nous référerons à ce mécanisme en tant que l'hypothèse du *monde gris*. Et, son application implique de choisir une zone ou la totalité de l'image pour obtenir la moyenne statistique des composantes rouge R_s, verte G_s et bleue B_s. Les gains de chacun des plans sont estimables de deux manières différentes :

1. la première consiste à choisir l'une des trois couleurs comme étant celle de référence. En pratique, il est conseillé de choisir le vert. Rappelons que dans la mosaïque *Bayer*, la composante verte contient 50% de l'information. Le modèle de Von Kries est applicable si on considère une relation linéaire entre les cônes et les réponses chromatiques sur les images, *i.e.*, $\boldsymbol{K}_L \approx \boldsymbol{K}_R, \boldsymbol{K}_M \approx \boldsymbol{K}_G$ et $\boldsymbol{K}_S \approx \boldsymbol{K}_B$, où :

$$\boldsymbol{K}_R = \frac{G_s}{R_s}\,, \quad \boldsymbol{K}_G = 1.0\,, \quad \boldsymbol{K}_B = \frac{G_s}{B_s}\,, \tag{2.21}$$

2. la seconde considère le niveau d'intensité d'une zone choisie comme paramètre de référence. Les gains deviennent alors :

$$\boldsymbol{K}_R = \frac{I_s}{R_s}\,, \quad \boldsymbol{K}_G = \frac{I_s}{G_s}\,, \quad \boldsymbol{K}_B = \frac{I_s}{B_s}\,, \tag{2.22}$$

où $I_s = (R_s + G_s + B_s)/3$. Sous ces conditions la moyenne des gains est égale à l'unité ce qui est recommandé lors de l'utilisation d'une méthode d'adaptation chromatique plus générale.

Ce mécanisme de correction est une composante importante du processus d'adaptation. Cependant, s'il est utilisé seul, la constance de couleur peut donner des résultats incorrects, typiquement quand l'hypothèse du monde gris n'est pas respectée [Kehtarnavaz 02] . Par exemple, une image de la forêt amazonienne contiendra un tel niveau de vert que l'utilisation d'une correction basé sur l'hypothèse du monde gris détruira la richesse chromatique d'origine : des images grisonnées sont alors obtenues.

L'égalisation automatique de couleur a été conçue principalement pour le perfectionnement des images numériques, caractérisées par une gamme dynamique de longueurs d'onde inférieure, comparée à la gamme élevée des vraies scènes du monde. Elle permet d'utiliser un modèle simplifié du système visuel humain complexe, et de reproduire qualitativement son mécanisme d'adaptation. Pour cette raison, une évaluation quantitative du modèle n'a pas été effectuée jusqu'ici, ce qui reste un domaine de recherche intéressant.

2.4.3.2 Hypothèse du *monde blanc*

Dans certains cas le système visuel humain normalise les réponses de ses photorécepteurs, en les maximisant vers une zone blanche hypothétique de référence, réalisant la constance de couleur. Nous nous référerons à ce mécanisme en tant que *hypothèse du monde blanc*. Dans le domaine du traitement d'images, ce processus est lié au problème de détection de la zone blanche de référence et n'est pas globalement très différent de l'approche de la section précédente.

Cette méthode permet d'obtenir les moyennes des composantes chromatiques, afin de ramener la zone de référence vers la valeur du « blanc » choisie (R_w, G_w, B_w). En photographie, il était courant de placer sur la scène un élément dit blanc pour retoucher les images acquises ultérieurement, la correction chromatique étant obtenue par la mise à l'échelle des trois plans couleur utilisant les équations suivantes :

$$\boldsymbol{K}_R = \frac{R_w}{R_s}, \quad \boldsymbol{K}_G = \frac{G_w}{G_s}, \quad \boldsymbol{K}_B = \frac{B_w}{B_s}. \tag{2.23}$$

Nous avons détecté que l'utilisation d'un objet blanc (un carton, une feuille en papier, ...) sur certaines caméras (surtout avec un contrôle automatique de gain activé), était un inconvénient car la couleur résultante sur l'objet ne témoignait d'aucune déviation de couleur. C'est à dire que dans les cas d'illumination intense de l'objet de référence, les photorécepteurs de la caméra saturent et la région devient blanche à la sortie. Il est commun que les caméras numériques utilisent la zone plus claire de l'image comme le blanc et la plus obscure comme le noir. Nous avons donc estimé plus raisonnable d'utiliser un objet gris, ou une grille de plusieurs tons de gris. En fait, l'utilisation d'une référence en tons de gris est un cas particulier de l'application de la technique de la section 2.4.3.1 qui fonctionne assez bien dans beaucoup de situations.

Ces approches fondées sur les hypothèses du monde gris ou blanc, nécessitent d'identifier dans la scène une zone sur laquelle les gains seront calculés. Sauf exploitation d'une zone prévue à cet effet pour une caméra fixe (contexte de la vidéo-surveillance), ce n'est pas compatible avec notre application de robotique mobile. Les applications robotiques sur environnements dynamiques imposent l'utilisation de techniques automatiques de détection du voile de couleur dominante et d'un équilibrage de la couleur en ligne. Deux de ces méthodes sont présentées dans les sections 2.4.3.3 et 2.4.3.4.

2.4.3.3 Retinex

La plupart des algorithmes de correction chromatique utilisent des changement d'illumination globaux dans l'image pour estimer les paramètres de correction [Bianco 02] tandis qu'en pratique,

très souvent, les variations sont locales. Par exemple, l'éclairage d'une salle blanche est affecté par la couleur de la lumière locale réfléchie par les objets (réflexion dépendant de la taille, de la couleur et du type de surface de l'objet) sur la scène.

L'ensemble d'algorithmes appelé *retinex* [Ciurea 04, Funt 04] proposé par Edwin H. Land fondateur de la marque *Polaroid*, prend en compte les relations locales d'illumination pour essayer d'imiter les mécanismes d'adaptation chromatique de la vision humaine [Land 71]. Ces mécanismes sont responsables du développement des techniques de constance de couleur et du perfectionnement de la gamme dynamique de tons sur l'image. Land a montré que les mécanismes de la vision ne sont pas des phénomènes ponctuels et bien localisés mais plutôt des phénomènes complexes qui considèrent l'interdépendance d'un ensemble ou d'un voisinage de photorécepteurs. Autrement dit, la couleur d'un seul point sur une image ne dépend pas de la lumière réfléchie par ce même point et lui seul, mais de la résultante des lumières réfléchies par ses points voisins et lui même. Les premières expériences avec cette méthode, ont exploité des affiches ou transparents de différentes couleurs projetées sur un écran. Les images résultantes s'appellent des *Mondrians* .

L'algorithme retinex basiquement commence par calculer la valeur de la luminance $\boldsymbol{L}_p$ sur un point $\boldsymbol{x}_p$ influencé par un ensemble de N points $\boldsymbol{x}_i$ choisis aléatoirement dans son voisinage, par la formule suivante :

$$\boldsymbol{L}_p = \frac{1}{N}\sum_{i=1}^{N}(log(I(\boldsymbol{x}_p)) - log(I(\boldsymbol{x}_i))) \tag{2.24}$$

l'opération est exécutée indépendamment sur toutes les composantes chromatiques (R,G,B). Par contre, ce processus conceptuellement simple est particulièrement complexe à mettre au point [Ciurea 04] puisqu'en réalité il considère beaucoup plus de paramètres que ceux exprimés par l'équation 2.24. Mais une fois qu'on arrive à le mettre au point, il est très performant.

Pour des applications en robotique un tel algorithme est difficilement acceptable, du fait des contraintes de temps réel qui rendent son exploitation prohibitive. Ainsi, les algorithmes pratiques utilisent une approche globale pour estimer en ligne la correction des déséquilibres chromatiques, afin de s'approcher de la constance de couleur.

2.4.3.4 Détection automatique de la couleur dominante

Nous allons décvrire finalement l'approche que nous avons développée pour l'identification automatique de la couleur dominante sur des images numériques. Cette approche est inspirée des travaux de F. Gasparini [Gasparini 03, Gasparini 04] sur la correction chromatique en photographie.

Le processus de détection commence par un sous échantillonnage de l'image d'origine car nous devons nous adapter aux contraintes d'une application en temps réel. Nous utilisons pour ceci des fréquences d'échantillonnage de 8, 16 et 32 sur des images de résolution pleine de 1024×768. Puisque l'un des buts consiste à obtenir des images agréables et d'apparence naturelle servant à caractériser l'impression visuelle, il est nécessaire d'utiliser un espace de couleur perceptuel adapté au système visuel, tel que $CIE - L^*ab$. Cet espace permet de séparer les composantes de luminance et de chromaticité. La conversion de l'espace RGB vers l'espace perceptuel L^*ab est rappelée en Annexe A : elle nécessite les équations A.2 pour passer de l'espace RGB à l'espace XYZ, puis, les équations A.11 pour calculer les coordonnées L^*ab.

Pour la détection automatique d'un déséquilibre chromatique, nous partons de l'hypothèse qu'une couleur dominante globale se manifeste généralement par la présence d'un pic ou mode principal sur un histogramme bidimensionnel $h_{ab}(k,l)$:

$$h_{ab}(i,j) = \frac{\#\{L^*(i,j)|i \in a, j \in b\}}{N_{ab}}\,, \tag{2.25}$$

où N_{ab} est le nombre des pixels sur le plan couleur ab. L'objectif consiste à détecter ce mode principal et étudier ses propriétés statistiques qui nous indiqueront la présence d'un possible déséquilibre chromatique. Les relations statistiques que nous avons utilisées sont données par les formules suivantes :

$$\begin{aligned} \mu_{i,j} &= \sum_{k=0}^{K}\sum_{l=0}^{L} k^i l^j h_{ab}(k,l) \\ \sigma^2_{i,j} &= \sum_{k=0}^{m}\sum_{l=0}^{L} (k-\widetilde{\boldsymbol{\mu_a}})^i (l-\widetilde{\boldsymbol{\mu_b}})^j h_{ab}(k,l) \end{aligned} \tag{2.26}$$

où $\widetilde{\boldsymbol{\mu_a}} = \mu_{01}/\mu_{00}$ et $\widetilde{\boldsymbol{\mu_b}} = \mu_{10}/\mu_{00}$. En utilisant ces formules, il est possible d'obtenir un paramètre appelé le cercle équivalent [Gasparini 04], qui est caractérisé par un centre localisé sur $C(\mu_a, \mu_b)$ et un rayon $\sigma = \sqrt{\sigma_{01}/\mu_{00} + \sigma_{10}/\mu_{00}}$. Ce cercle représente le degré de dispersion des pixels autour de la couleur dominante.

Dans notre implémentation un classifieur est défini par deux termes ; le premier est donné par l'équation $D_\sigma = \sqrt{\mu_a^2 + \mu_b^2}/\sigma - 1$.

Le second se calcule par les mêmes statistiques mais centrées sur (μ_a, μ_b), le rayon σ se réduisant jusqu'à contenir un pourcentage important des pixels (40%) de l'image, il est donné par $D'_\sigma = \sqrt{\mu_a'^2 + \mu_b'^2}/\sigma' - 1$.

La présence d'une couleur dominante (à corriger) est établie si la condition suivante est vérifiée : $D_\sigma + D'_\sigma > 1.0$.

Pour obtenir les éléments responsables du déséquilibre chromatique, la valeur maximale de luminance L^*_{max} doit être calculée. Cela établit l'intervalle des valeurs pertinentes pour notre analyse ($0.35L^*_{max} \leq L^* \leq 0.95L^*_{max}$). Il est conseillé d'éviter les valeurs trop claires ou trop sombres qui représentent parfois les réponses saturées du capteur. Nous calculons ainsi les moyennes chromatiques en R, G, B à l'intérieur du cercle de rayon σ', qui nous permettront d'obtenir les coefficients de correction par la mise à l'échelle utilisant la formule 2.23. En principe, les coefficients obtenus vont corriger chromatiquement l'image en affectant la valeur des pixels dans l'image d'origine (*cf.* équation 2.20).

Un inconvénient de cette détection, non prise en compte dans nos travaux, est qu'il est nécessaire de vérifier que la couleur dominante détectée (liée aux conditions d'éclairage ou aux caractéristiques intrinsèques du capteur) ne soit pas une couleur qui domine de manière naturelle dans l'image (*cf.* l'exemple précédent d'une photo d'une forêt). Il serait nécessaire dans l'absolu, de construire un classifieur sur le type de scène observée par la caméra, classifieur qui nous permettrait d'identifier globalement lorsqu'une image a besoin d'une correction couleur ; c'est un sujet qui paraît complexe.

2.4.4 Modèles d'adaptation chromatique

Les modèles d'adaptation chromatique sont utilisés dans les systèmes de traitement d'images couleur, pour transformer l'apparence colorimétrique de l'image en fonction de différentes sources d'illumination. L'adaptation chromatique peut être appréciée par un observateur qui examine un objet dit « blanc » sous l'influence de différents types d'illumination comme la lumière du jour ou l'incandescence ; malgré ces variations d'illumination, l'objet « blanc » doit maintenir son apparence blanche sous les deux sources lumineuses, dès que l'observateur est adapté à la source d'illumination [Bianco 02]. De manière plus générale, les spectres réfléchis par un objet à midi et le soir doivent nous donner la même impression colorée[Reinhard 01].

Les méthodes d'adaptation chromatique suivantes, sont très souvent référencées dans la littérature pour la caractérisation de caméras numériques et l'échange d'information : Von Kries, Nayatani, RLAB [Fairchild 96], Bradford, Sharp [Ward 02] et CMCCAT2002 [Süsstrunk 01]. Il est important de préciser que la plupart de ces méthodes se fondent sur l'hypothèse de Von Kries.

La transformation chromatique appliquée aux valeurs primaires du tri-stimuli (X', Y', Z') d'une couleur acquise en présence d'une source lumineuse A, prédit les valeurs du tri-stimuli (X'',Y'', Z'') de cette même couleur avec une autre source de lumière B.

Les valeurs primaires du tri-stimuli (X', Y', Z') sont d'abord transformées linéairement en utilisant une matrice M de dimension 3x3, vers les réponses des cônes L, M, S. Cette matrice va modifier les réponses de l'adaptation des cônes sous la première source lumineuse A (R', G', B'). Les couleurs obtenues (R', G', B') sont mises à l'échelle de manière indépendante en obtenant, ainsi la réponse d'adaptation (R'', G'', B'') sur la seconde source B. En absence de coefficients non linéaires dans la matrice de changement d'échelle, cette transformation peut être exprimée par une matrice diagonale. Enfin, les valeurs prédites du tri-stimuli (X'',Y'', Z'') sont obtenues à partir des réponses des cônes, par la transformation inverse M^{-1}.

L'équation 2.27 qui décrit les relations mathématiques décrites précédemment, est connue comme le modèle d'adaptation chromatique linéaire[15],

$$\begin{pmatrix} X'' \\ Y'' \\ Z'' \end{pmatrix} = M^{-1} \begin{pmatrix} R''_w/R'_w & 0 & 0 \\ 0 & G''_w/G'_w & 0 \\ 0 & 0 & B''_w/B'_w \end{pmatrix} M \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} \tag{2.27}$$

Le choix de la matrice M établit, certainement, la différence fondamentale entre les différentes méthodes d'adaptation chromatique basées sur l'hypothèse de Von Kries. Il est clair que le modèle de l'équation 2.27 est une généralisation du modèle initialement proposé, d'une simple mise à l'échelle(équation 2.20) pour corriger le déséquilibre chromatique.

2.4.4.1 Matrices de transformation chromatique

Johannes Von Kries a émis l'hypothèse que l'adaptation chromatique du système visuel humain aux conditions d'éclairage variables pouvait être modélisée par un contrôle indépendant des gains sur les cônes de type L, M et S. Ainsi, la matrice de changement d'échelle (matrice diagonale dans l'équation 2.27) est construite à partir du rapport des réponses obtenues par les cônes dans des conditions d'éclairage différentes.

[15] Dans [Fairchild 98] est présentée une description détaillée des modèles d'adaptation chromatique non linéaires

Considérant cette hypothèse, la matrice M de l'équation 2.27 doit transformer linéairement les valeurs primaires du tri-stimuli X, Y, Z vers les réponses des cônes L, M, S. Cette matrice, estimée expérimentalement par Hunt, Pointer and Estevez [Katoh 01] est donnée par :

$$M_{HPE} = \begin{pmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \tag{2.28}$$

À l'université de Bradford, Lan [Finlayson 00] a obtenu une autre matrice à partir d'expériences sur la constance de couleur (en étudiant l'apparence d'objets variés richement colorés soumis à plusieurs sources d'illumination (A et D65)). Cette transformation a une correction non-linéaire sur la bande bleue mais en pratique on utilise très souvent la version linéaire donnée par :

$$M_{BFD} = \begin{pmatrix} 0.8951 & 0.2664 & -0.1614 \\ -0.7502 & 1.7135 & 0.0367 \\ 0.0389 & -0.0685 & 1.0296 \end{pmatrix} \tag{2.29}$$

L'utilisation de cette transformation implique que la correction ne se réalise pas exactement dans l'espace des cônes mais dans un espace plus étroit et sensitif. Ainsi, les sensibilités aux longueurs d'onde larges et médium sont davantage décorrélées.

Une modification à la matrice de Bradford qui théoriquement devrait nous permettre de projeter sur un espace des cônes plus net, est représentée sur l'équation suivante :

$$M_{Sharp} = \begin{pmatrix} 1.2694 & -0.0988 & -0.1706 \\ -0.8364 & 1.8006 & 0.0357 \\ 0.0297 & -0.0315 & 1.0018 \end{pmatrix} \tag{2.30}$$

La dernière transformation étudiée est la matrice du système CMCCAT2000 qui calcule le niveau d'adaptation de façon différente des systèmes dérivés de la matrice de Bradford,

$$M_{CMC} = \begin{pmatrix} 0.7982 & 0.3389 & -0.1371 \\ -0.5918 & 1.5512 & 0.0406 \\ 0.0008 & 0.0239 & 0.9753 \end{pmatrix} \tag{2.31}$$

Ceci est l'une des dernières transformations dites linéaires car la tendance actuelle considère plutôt l'utilisation des méthodologies basées sur des mécanismes non-linéaires comme la transformation RLAB [Fairchild 96].

2.4.4.2 Espaces de couleur numérique RGB

La plupart des applications d'échanges d'informations numériques utilisent l'espace RGB (soit à la sortie des caméras, soit à l'entrée des écrans) pour la représentation générale d'images. Ces espaces peuvent être considérés comme indépendants du point blanc (white-point) car les valeurs RGB sont le résultat de la post-adaptation de cônes. Dans ces situations où les valeurs R, G, B sont l'espace de travail, une matrice de conversion $RGB \rightarrow XYZ$, doit être utilisée pour obtenir les valeurs primaires du tri-stimuli.

La matrice standard est fournie en Annexe A. Mais, en fait, plusieurs conversions entre espace CIE XYZ et RGB ont été proposées, car plusieurs espaces RGB sont proposés à l'utilisation selon les équipements : sRGB (standard RGB), ROMM (Reference Output Medium Metric) et Prime RGB. Ce dernier espace est basé sur des longueurs d'onde de 450, 540 et 605nm ; les moniteurs

basés sur ces longueurs d'onde, produisent un gamut plus répandu et sont visuellement très efficaces. La transformation Prime RGB est donnée par :

$$M_{prime} = \begin{pmatrix} 2.0016 & -0.5576 & -0.4440 \\ -0.7997 & 1.6627 & 0.1371 \\ 0.0089 & -0.0190 & 1.0100 \end{pmatrix} \tag{2.32}$$

Le standard ROMM a été développé par *Eastman Kodak* dans le cadre de la manipulation, de l'édition et du rendu d'images codées sur l'espace RGB. C'est un espace RGB de large gamut qui n'est attaché à aucun type d'écran,

$$M_{ROMM} = \begin{pmatrix} 1.2977 & -0.2556 & -0.0422 \\ -0.5251 & 1.5082 & 0.0169 \\ 0.0000 & 0.0000 & 1.0000 \end{pmatrix} \tag{2.33}$$

Enfin, le dernier standard pour l'échange d'informations en télévision de haute définition, le sRGB (aussi appelé ITU-R BT.709 [Plataniotis 00]) est adapté pour la majorité des écrans à tube cathodique munis d'une couche de phosphore.

$$M_{sRGB} = \begin{pmatrix} 2.564586 & -1.163667 & -0.413517 \\ -1.021910 & 1.977611 & 0.045616 \\ 0.078198 & -0.263760 & 1.233257 \end{pmatrix} \tag{2.34}$$

2.4.4.3 Correction chromatique utilisant une CAT

Pour appliquer les transformations d'adaptation chromatique sur des systèmes qui travaillent sur des représentations RGB, il faut d'abord faire la transformation de ces valeurs vers l'espace CIE-XYZ utilisant l'une des transformations de la section 2.4.4.2. La suite consiste à utiliser l'équation 2.27 qui modélise le processus d'adaptation chromatique. En pratique, les matrices de conversion peuvent être concaténées vers une forme plus compacte,

$$\begin{aligned} \begin{pmatrix} R'' \\ G'' \\ B'' \end{pmatrix} &= M_D M_T^{-1} \begin{pmatrix} R''_w/R'_w & 0 & 0 \\ 0 & G''_w/G'_w & 0 \\ 0 & 0 & B''_w/B'_w \end{pmatrix} M_T M_D^{-1} \begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} \\ &= M_{CAT} \begin{pmatrix} \boldsymbol{K}_R & 0 & 0 \\ 0 & \boldsymbol{K}_G & 0 \\ 0 & 0 & \boldsymbol{K}_B \end{pmatrix} M_{CAT}^{-1} \end{aligned} \tag{2.35}$$

M_T est une des matrices de transformation de l'espace CIE XYZ vers l'espace des cônes, présentée en section 2.4.4.1. M_D est une des matrices présentée en section 2.4.4.2, pour la conversion de CIE XYZ vers l'espace de représentation RGB des données et de l'affichage. Les éléments de la matrice diagonale d'échelle sont remplacés par les gains de la correction chromatique souhaitée. Nous avons détecté que la sélection des éléments de la matrice de mise à l'échelle n'est pas complètement arbitraire, elle doit être équilibrée[16] ($\sum_i K_i^{-1} = 3$ où $i \in \{R_w, G_w, B_w\}$) pour éviter de trop atténuer ou trop renforcer l'une des composantes chromatiques.

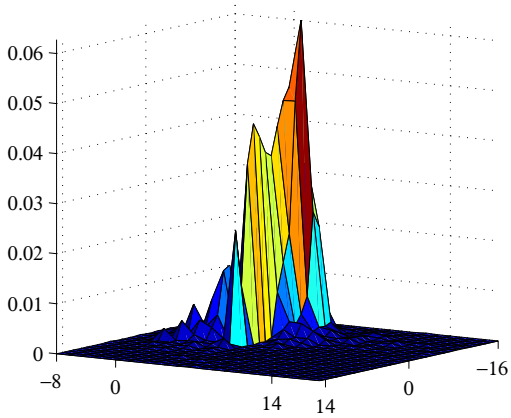
Il reste à sélectionner la transformation la mieux adaptée aux propriétés chromatiques intrinsèques de notre capteur, propriétés inconnues et très compliquées à obtenir expérimentalement

[16] Nous avons testé des autres expressions donnant des résultats équivalents, *i.e.*, $\prod_i K_i^{-1} = 1$ où $K_i = \sqrt[3]{R_w G_w B_w}/i$ et $i \in \{R_w, G_w, B_w\}$

(a) Image démosaïquée

(b) Image calibrée chromatiquement

(c) Agrandissement de la région de référence



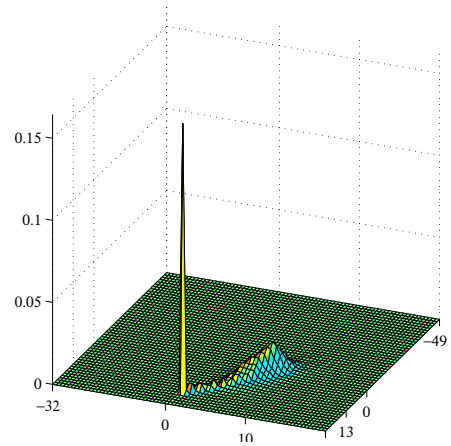
(d) Histogramme du déséquilibre chromatique

FIG. 2.8 – Analyse quantitative de la balance des blancs par histogramme

sans l'instrumentation adéquate (colorimètre ou spectrophotocolorimètre) [Ohno 97, Ward 02]. Pour cette sélection, il existe plusieurs possibilités dont celle de Hasler [Hasler 04] qui considère par exemple que dans le cas où la transformation linéaire appliquée dans la caméra (capteur) est inconnue, la matrice identité est une estimation raisonnable (justifiée si des opérations de linéarisation ont été effectuées lors de l'acquisition) pour la matrice M_{CAT}. Dans notre cas, nous utilisons un critère statistique et la connaissance a priori d'une région « blanche » (*i.e.*, R=G=B) dans l'image pour mesurer le comportement de plusieurs matrices M_{CAT} sur notre système et choisir celle qui est la plus adaptée à notre capteur.

Pour évaluer une matrice M_{CAT}, nous considérons que l'histogramme couleur (espace L^*ab) calculé sur la région de référence de l'image corrigée chromatiquement, doit être centré à l'origine et que moins celui-ci est étalé, meilleure est la correction. La figure 2.8 présente l'image d'origine, sa correction chromatique, une portion de l'image contenant la région de référence et l'histogramme couleur de cette région. Cet histogramme permet d'obtenir des critères statistiques pour identifier le meilleur ensemble de matrices de transformation chromatique. Les

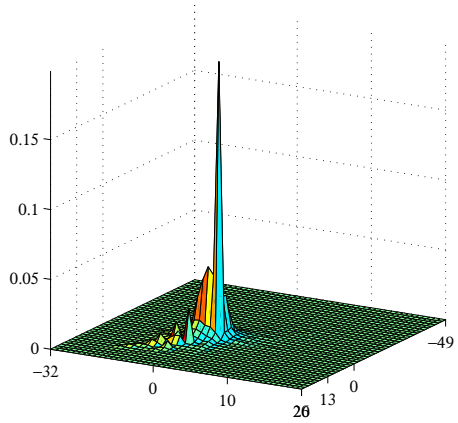
(a) Image demosaïquée



(b) Histogramme L^*ab sur la zone de référence

FIG. 2.9 – Déséquilibre chromatique sur l'image d'origine

(a) Image corrigée (mise à l'échelle)



(b) Histogramme L^*ab sur la zone de référence

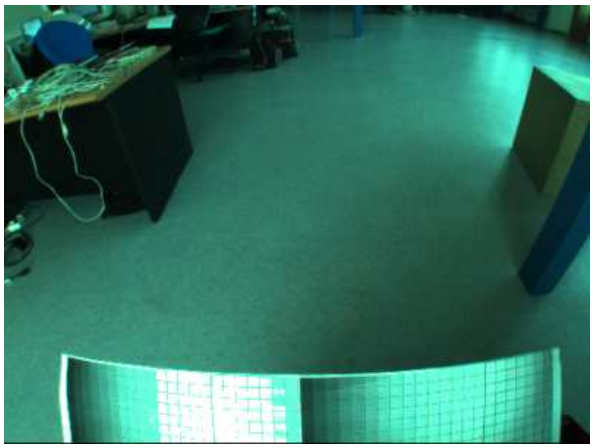
FIG. 2.10 – Déséquilibre chromatique sur l'image corrigée par l'hypothèse de Von Kries

critères de sélection reposent sur la distance du centroïde (μ_{ab}) de la distribution à l'origine et la dispersion de la distribution (σ_{ab}).

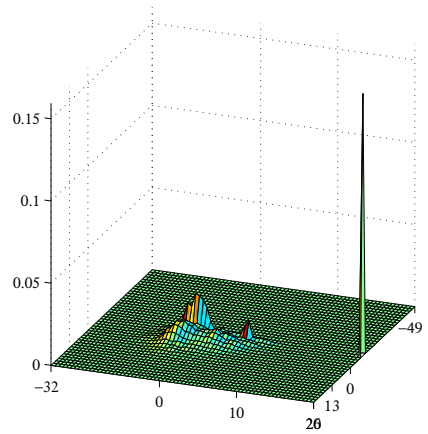
Parmi toutes les matrices de conversion (affichage) $RGB \rightarrow XYZ$, les meilleures sont la *ROMM* et la *PRIME* et les pires sont la *RGB* et la *sRGB* pour notre capteur. Nous avons donc choisi la matrice *ROMM* pour réaliser les tests présentés sur les figures 2.9-2.14.

La couleur dominante sur l'image d'origine est facilement visible sur l'histogramme de la figure 2.9(b) car le mode principal de cette distribution est bien concentré et éloigné de l'origine.

Nous détectons que la matrice de Hunt-Pointer-Estevez présente une forte déviation de gamut vers le vert ce qui détériore la correction chromatique. Ceci est constaté par son mode principal qui est localisé loin de l'origine sur l'histogramme de la figure 2.11. Par contre, les matrices de Bradford, *Sharp* et *CMCC200* donnent des résultats perceptuellement similaires.



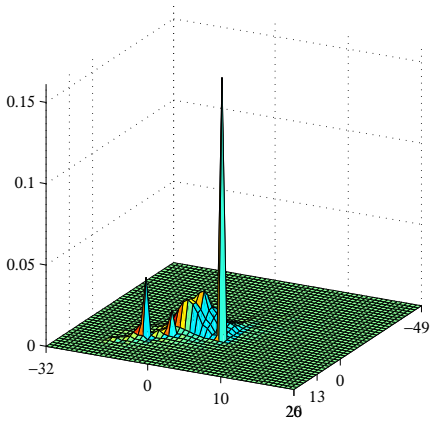
(a) Image corrigée (Hunt-Pointer-Estevez)



(b) Histogramme L^*ab sur la zone de référence

FIG. 2.11 – Déséquilibre chromatique sur l'image, corrigée par les matrices Hunt-Pointer-Estevez et *ROMM*

(a) Image corrigée (Bradford)



(b) Histogramme L^*ab sur la zone de référence

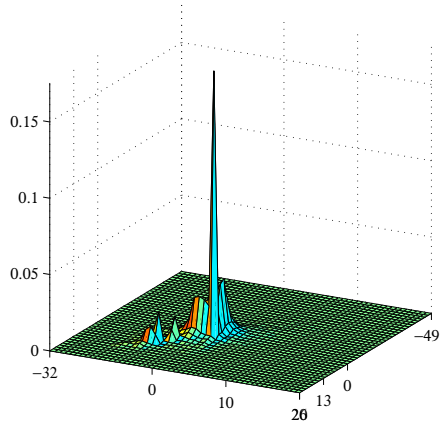
FIG. 2.12 – Déséquilibre chromatique sur l'image corrigée par les matrices Bradford et *ROMM*

Numériquement les statistiques nous indiquent que la matrice *Sharp* a en général corrigé le mieux les déséquilibres chromatiques sur notre capteur.

Notons toutefois que le modèle de base (simple mise à l'échelle sur l'espace RGB, M_{CAT} matrice identité) à trois paramètres (voir figure 2.10) a donné d'excellents résultats parfois aussi bons qu'avec la combinaison des matrices *Sharp* et *ROMM*.

De ce fait, pour fournir une adaptation chromatique fonction de l'observateur et éviter des problèmes de couleur dominante dans nos images, nous appliquons simplement une transformation linéaire du type Von Kries (équation 2.20)

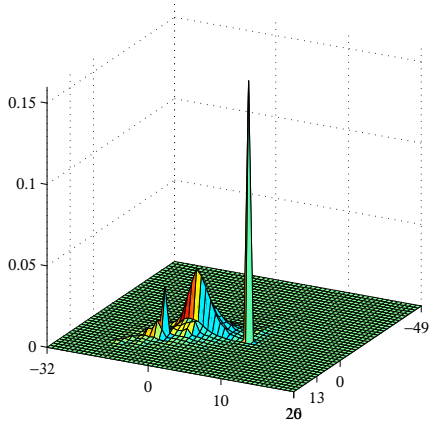
(a) Image corrigée (*Sharp*)



(b) Histogramme L^*ab sur la zone de référence

FIG. 2.13 – Déséquilibre chromatique sur l'image corrigée par les matrices *Sharp* et *ROMM*

(a) Image corrigée (*CMCC2000*)



(b) Histogramme L^*ab sur la zone de référence

FIG. 2.14 – Déséquilibre chromatique sur l'image corrigée par les matrices *CMC2000* et *ROMM*

2.5 Correction gamma

Dans le contexte du traitement d'images, de la conception assistée par ordinateur, de la vidéo et de la photographie numérique, le symbole γ représente un paramètre qui décrit la non-linéarité de l'intensité dans la reproduction. Le tube cathodique (CRT) employé dans les écrans a un comportement non-linéaire dans le sens où l'intensité de la lumière reproduite à la sortie de l'afficheur n'est pas une fonction linéaire du voltage d'entrée [Katajamäki 03]. La réponse du CRT suit une loi de puissance en fonction du voltage,

$$C_o(x, y) = (v(x, y))^{\gamma} \tag{2.36}$$

Par ailleurs, le système visuel humain a un comportement perceptuel non-linéaire vis à vis de

l'intensité. Il est quasiment logarithmique et son comportement est presque une fonction inverse de celui du tube cathodique.

Dans le contexte de l'amélioration et de l'analyse automatique d'images, la valeur γ peut avoir un effet crucial. Elle établit la façon dont les régions obscures ressortent par rapport aux détails des régions plus illuminées. En conséquence, elle affecte aussi la saturation, et en moindre degré la teinte (tonalité) [Siebert 01, Lauzière 99].

La valeur γ la plus adaptée est celle qui donne la meilleure concordance entre l'image imprimée, ou l'image photographique sur support papier de qualité, et cette même image observée à l'écran.

Avec cette valeur γ la plus adaptée, une fonction inverse $x^{1/\gamma}$ inverse sur la luminance en chaque pixel peut être utilisée, avec un élément de correction :

$$C_o(x,y) = C_{max}\left(\frac{C_i(x,y)/C_{max} + 0,055}{1,055}\right)^{1/\gamma} \tag{2.37}$$

où $C_i(x,y)$ est la valeur du pixel à l'entrée, C_{max} est la valeur maximale atteinte à la sortie (un choix de 255 donne la plage dynamique la plus grande à l'image). Sur un écran, même pour une valeur d'entrée égale à zéro, on observe parfois une faible luminance (luminosité résiduelle), ce qui est pris en compte dans l'équation précédente par la valeur 0,055.

L'opération d'appliquer une fonction inverse $x^{1/\gamma}$ aux données avant l'affichage est connue sous le nom *correction gamma* de l'image. La valeur γ peut osciller pour un écran entre 2.3 et 2.6, mais la valeur 2,2 est considèrée comme standard pour beaucoup d'écrans. En particulier, le standard ITU-R BT.709 recommande une correction de $\gamma = 2,2$ (*i.e.*, $1/\gamma = 0,45$) sur les systèmes de télévision de haute définition (évidemment dans des situations où la valeur exacte est inconnue).

2.5.1 Application du facteur gamma

Nous avons appliqué cette correction de deux façons différentes :

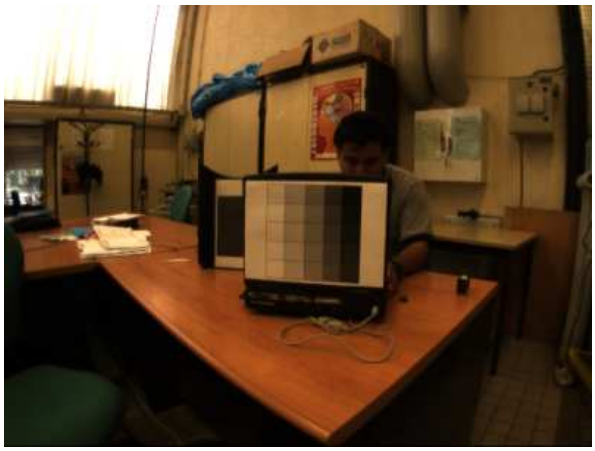
- ⋆ la première, en appliquant individuellement la correction 2.37 sur chacun des plans (R,G,B). Ceci nous permet de nous servir des valeurs de gamma distinctes (*i.e.*, $\boldsymbol{\gamma}_R$,$\boldsymbol{\gamma}_G$ et $\boldsymbol{\gamma}_B$,) pour gérer soit la correction gamma sur chaque composante, soit une correction de balance de blancs[17].
- ⋆ la seconde se sert de la correction gamma appliquée sur l'intensité pour rehausser les composantes chromatiques, ce qui est formulé de la manière suivante :

$$C_o^x = C_i^x \cdot (C_i)^{1/\gamma - 1} \tag{2.38}$$

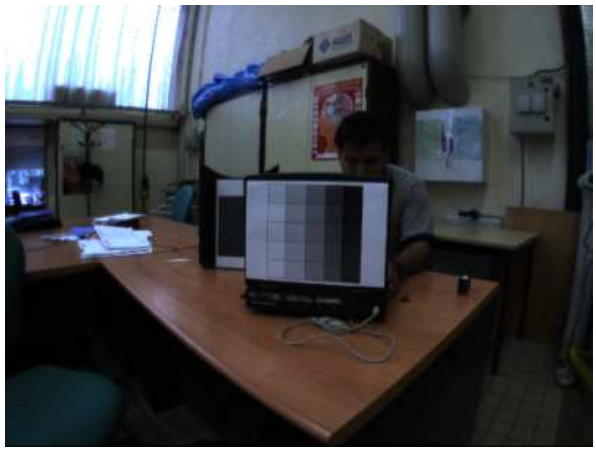
où C_i^x représente l'une des composantes chromatiques (R,G,B) et C_i leur intensité avant correction ($\sum_{x=1}^{3} C_i^x/3$).

La figure 2.15 montre l'effet de la correction gamma sur des images qui ont une gamme dynamique limitée. Les images 2.15 (c) et (d) ont été corrigées avec un facteur gamma de 2,2 : en (c), une correction individuelle a été appliquée sur chacune des bandes ; en (d), une correction sur l'intensité a modulé l'amplitude de chacune des composantes chromatiques. Une amélioration de l'impression visuelle est perçue dans les deux cas : en (c), on obtient un éclaircissement de l'image et plusieurs détails sont bien visibles, surtout dans les zones obscures ; en (d), on perçoit

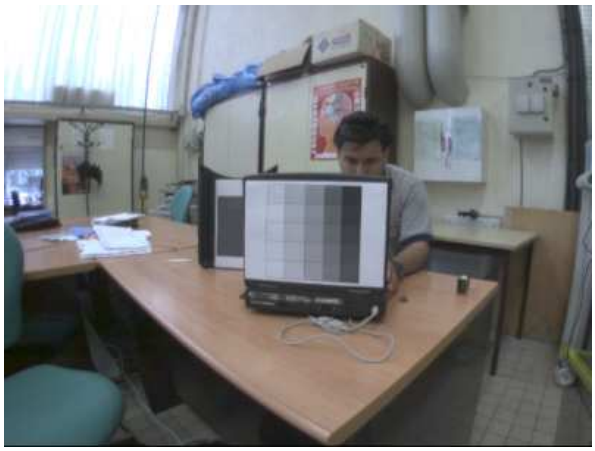
[17]Notons que la correction des blancs utilise une formulation avec une puissance de γ_x et que la correction gamma proprement dit utilise une correction de $1/\gamma$



(a) Image démosaïque



(b) Correction chromatique



(c) Correction gamma sur chaque composante

(d) Correction gamma sur l'intensité

FIG. 2.15 – L'effet de la correction gamma sur l'apparence de l'image. Le facteur gamma pour les images (c) et (d) est fixé à 2,2

aussi une agréable dynamique des couleurs et généralement ce sont des couleurs *plus vives* (*hot colors*) qui dominent la scène. C'est cette correction qui donne la meilleure impression visuelle.

Nous avons développé un algorithme basé sur histogrammes pour détecter quelles images ont besoin d'une correction gamma et de combien doit être la correction requise. Nous avons essayé de mesurer l'énergie sur l'histogramme en appliquant uniformément plusieurs valeurs de gamma : dans la plupart des cas, ce test nous indiquait qu'une correction gamma n'était pas nécessaire, ce qui n'était pas toujours vrai. Un autre critère devrait donc être étudié.

2.5.2 Correction chromatique par courbes gamma

Nous avons testé une approche de correction chromatique utilisant des courbes non linéaires exploitant aussi un facteur gamma $I(x,y) = (I_o(x,y))^{1/\gamma}$. Pour cela, nous utilisons une grille monochromatique de tons de gris, l'objectif consistant à équilibrer la couleur employant une

courbe de correction indépendante sur chaque plan $\boldsymbol{I}_R, \boldsymbol{I}_G, \boldsymbol{I}_B$. Le problème consiste alors à estimer $\gamma_r, \gamma_g, \gamma_b$.

Généralement, par simplicité on fixe une valeur unitaire pour la composante verte de référence $\gamma_g = 1,0$. On cherche alors à estimer la relation entre deux variables x et y sur la forme d'une fonction f qui dépend du paramètre γ. Nous considérerons dans la suite le vert comme étant la variable indépendante, la correction sur le rouge ou le bleu étant déterminée à partir de la relation :

$$\hat{y}_k = f(x_k, \gamma) = a \left(\frac{x_i}{b}\right)^{\gamma} \tag{2.39}$$

où a et b sont des facteurs d'échelle fixés à 255.

Les paramètres inconnus sont obtenus par une méthode de régression non-linéaire exploitée pour minimiser aux moindres carrés, la somme des carrés des écarts pondérés J :

$$J(\gamma) = \sum_{k=1}^{n} \boldsymbol{w}_k \left(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right) \tag{2.40}$$

L'estimation initiale du paramètre γ est typiquement fixée dans un intervalle $0,333 < \gamma < 3$ pour éviter des problèmes de convergence avec la classique méthode de Levenberg-Marquardt.

Cette méthode a donné des résultats approximatifs acceptables (voir figure 2.16), mais la correction n'est pas tout à fait fidèle à cause de la non linéarité de ce type de courbes. Les plus grosses erreurs sont visibles dans des pixels de luminance élevée.

Nous avons envisagé d'implémenter ce modèle en prenant en compte la nécessité d'un affichage des images sur des écrans qui généralement requièrent une compensation du facteur gamma. Nous avons constaté qu'il était assez avantageux d'utiliser une combinaison entre la correction chromatique et la correction gamma pour l'affichage. En effet, il est possible de représenter les deux paramètres par un seul facteur, par exemple pour le canal rouge nous aurions $R_d(x,y) = (R_o(x,y))^{\gamma_d/\gamma_r}$ avec γ_d pour corriger à la fois le voile chromatique et la non-linéarité du capteur, ce qui nous permet d'obtenir un affichage agréable de l'image.

2.6 Conclusion

Ce chapitre a présenté une chaîne de traitement pour l'acquisition d'images couleur de qualité optimale, à partir d'une caméra mono-CCD munie d'un filtre *Bayer*. Premièrement, nous avons donné les techniques les plus récurrentes dans la littérature pour la reconstruction de la couleur à partir des techniques d'interpolation ou de démosaïquage. La méthode de Hamilton est la principale technique que nous utilisons dans notre traitement.

La figure 2.16, illustre l'effet qu'un changement significatif dans l'illumination naturelle peut susciter dans les signaux de couleur. L'apparence de la couleur sur un objet varie donc radicalement, en fonction des conditions d'illumination, de la géométrie de la scène (surfaces de réflectivité)... Comment exploiter encore la couleur comme attribut intrinsèque d'un objet ?

Nous avons étudié quelques-unes des techniques pour détecter la présence d'une couleur dominante sur l'image obtenue après la reconstruction de la couleur. L'hypothèse de Von Kries, nous permet de corriger chromatiquement nos images en utilisant uniquement une estimation de la couleur qui provoque un tel déséquilibre. La méthode de Gasparini [Gasparini 03] nous a

(a) Image voilée par une couleur dominante

(b) Image corrigée par l'application d'un facteur gamma indépendant sur chaque bande

FIG. 2.16 – Image équilibrée chromatiquement par courbes de facteur gamma

permis d'estimer *on-line* les coefficients de correction chromatique induit par l'hypothèse de Von Kries, à l'aide de statistiques calculées sur l'image à corriger.

Nous avons aussi exploré les modèles d'adaptation chromatique, pour vérifier leur utilité dans le cadre de la correction et de la constance des couleurs. Cependant, nous avons détecté que malgré sa simplicité le modèle de Von Kries était parfois aussi performant que le meilleur modèle d'adaptation chromatique utilisé, ce qui dépend évidemment des caractéristiques intrinsèques du capteur.

Pour conclure sur cette difficile question de la constance des couleurs, citons Finlayson, qui a généralisé [Finlayson 95] le concept d'adaptation chromatique de J. Von Kries par la déclaration suivante : « si l'illumination et la réflectivité des objets sont régies respectivement par un monde fini de dimension 2×3, il existe donc une transformation sur le capteur dans laquelle une matrice diagonale supporte une constance à la couleur parfaite ».

Ainsi, l'efficacité du modèle diagonal (équation 2.35) dépend principalement des propriétés visuelles du capteur, notamment si les composantes chromatiques sont de bande passante étroite et surtout si elles ne se superposent pas. Sous cette hypothèse, nous considérons alors que notre système permet d'adapter l'image aux changements d'illumination dans certaines limites. Par contre, la constance de couleur est un domaine complexe qui est encore ouvert aux nouvelles théories pour mieux la comprendre.

Dans le chapitre suivant, nous décrivons comment une description $2D$ de la scène peut être construite, à partir d'une image couleur acquise par notre caméra mono-ccd et corrigée par les méthodes décrites ci-dessus.

Chapitre 3

Modélisation de l'environnement : détection des régions navigables

3.1 Introduction

Une étape fondamentale de la plupart des systèmes de vision par ordinateur est d'engendrer une description synthétique d'une image, plus exploitable que l'ensemble des pixels. Il n'est pas concevable pour un robot de réagir face au monde de manière autonome sans percevoir et identifier les principales composantes de l'environnement [Ghallab 97]. Rappelons que nos travaux concernent uniquement les déplacements d'un robot dans un environnement semi-structuré, plus spécialement dans un réseau de chemins. Dans ce contexte, notre robot autonome doit disposer d'une fonction de perception afin qu'il puisse, sans intervention humaine, déterminer à partir de l'image courante, où se trouvent les espaces navigables, notamment les *chemins*, les *croisements de chemins* et les entités spécifiques (arbres, bâtiments...) liés à la tâche qui doit être exécutée dans cet environnement.

Les descriptions synthétiques produites par cette fonction, seront ensuite exploitées lors de deux étapes principales, qui seront analysées dans les deux chapitres suivants :

- construire une représentation de l'environnement sans aucune connaissance a priori : ce modèle sera exploité, afin de planifier les actions du robot, et spécialement, les déplacements dans les espaces navigables ou sur le réseau de chemins ;
- et exécuter ces déplacements en identifiant chemins, espaces navigables et entités utiles à la tâche.

Dans ce mémoire, nous ne prenons en compte qu'un seul capteur embarqué sur le robot : une caméra couleur. De ce fait, la description de la scène courante est construite à partir d'une méthode de segmentation d'images couleur : c'est une étape cruciale pour réduire la quantité d'informations et représenter une scène par des primitives interprétables et facilement manipulables d'un point de vue informatique. La figure 3.1 illustre notre méthode, inspirée des travaux préliminaires de R. Murrieta Cid [Murrieta-Cid 98]. La description d'une scène à partir d'une seule image, est construite en quatre principales étapes :

- la segmentation de l'image, en exploitant uniquement la couleur ; l'état de l'art et notre méthode de segmentation d'images couleur, sont décrites en § 3.2.1 ;
- la caractérisation de chaque région extraite par un vecteur d'attributs (couleur, texture et information contextuelle) ; les différentes méthodes proposées pour quantifier la texture dans une région, sont d'abord décrites en § 3.3 ; l'ensemble des attributs sont ensuite

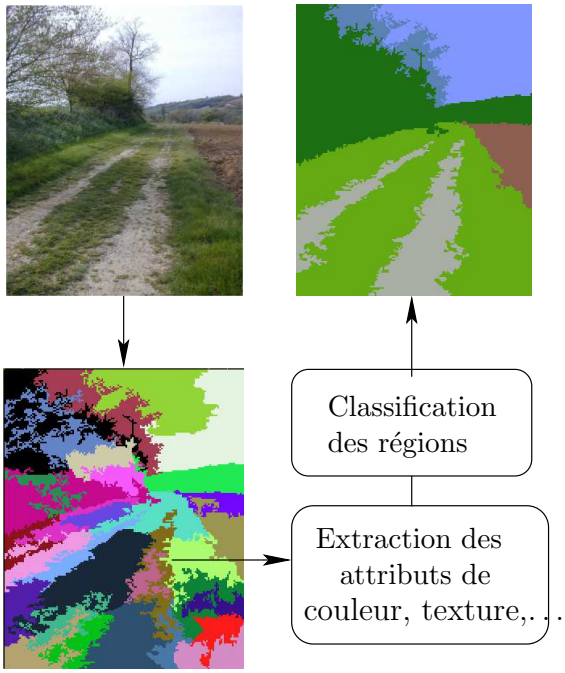


FIG. 3.1 – Les étapes de la modélisation 2D de scènes naturelles

présentés en § 3.4.
- la classification supervisée de ces régions ; nous décrirons tout d'abord en § 3.5 comment les vecteurs d'attributs caractéristiques pour les différentes classes à identifier sont appris sur une base d'images ; puis nous présenterons en § 3.6, les deux classifieurs testés pour identifier les régions.
- et finalement la fusion des régions voisines appartenant à la même classe.

La section § 3.7 présente des résultats expérimentaux, sous une forme similaire à la figure 3.1 : image d'origine, image des régions segmentées, image des régions étiquetées. Nous présentons enfin un bilan statistique sur les réussites et les échecs de notre approche afin d'identifier la nature des différentes régions extraites sur une image couleur acquise en milieu extérieur, notamment afin de reconnaître et localiser dans l'image le chemin sur lequel un robot doit se déplacer.

3.2 La segmentation couleur

3.2.1 Définition de la segmentation

La segmentation d'images est une opération fondamentale dans le domaine du traitement d'images : elle est l'étape critique dans de nombreuses méthodes de reconnaissance d'objets. Elle permet l'extraction des principales régions (éléments constitutifs) de la scène en utilisant un ou plusieurs critères d'homogénéité (couleur, texture, forme, ...) et de connexité spatiale [Krumm 95], *i.e.*, deux zones disjointes ayant les mêmes caractéristiques formeront des régions distinctes sur l'image segmentée.

Le but est que les régions correspondent à l'image d'entités présentes dans la scène, disjointes et de nature différente. Vis-à-vis de ce critère, on parle de sur-segmentation lorsqu'une même entité de la scène est fragmentée en plusieurs régions dans l'image, et de sous-segmentation lorsqu'au contraire, plusieurs entités de la scène n'ont pas pu être isolées, et donc se retrouvent

réunies dans une même région dans l'image. C'est une des opérations de base de traitement d'images, une des plus complexes sur laquelle peut reposer la qualité globale de tout un système de vision [Yan 03].

[Pal 93] a défini formellement la segmentation d'images de la manière suivante : soit $\mathcal{F}_I$ une image composée de pixels et $\mathcal{H}$ un prédicat d'homogénéité défini sur tout sous-ensemble de pixels ou régions connexes de $\mathcal{F}_I$. La segmentation de $\mathcal{F}_I$ est l'opération qui permet d'obtenir la partition $\mathcal{P}$ de $\mathcal{F}_I$ en N régions de pixels $\mathcal{R}_n$, $n = 1, \ldots, N$, telle que :

1. $\bigcup_{n=1}^{N} \mathcal{R}_n = \mathcal{F}_I$, avec $\mathcal{R}_n \cap \mathcal{R}_m = \emptyset$, $\forall n\, m(n \neq m)$,
2. $\mathcal{H}(\mathcal{R}_n) =$ Vrai, $\forall n$,
3. $\mathcal{H}(\mathcal{R}_n \cup \mathcal{R}_m) =$ Faux, $\forall n\, m$ avec $\mathcal{R}_n$ et $\mathcal{R}_m$ régions adjacentes .

La première condition exprime que la partition doit couvrir toute l'image, la seconde implique que chaque région doit être homogène d'après le prédicat $\mathcal{H}$ et la troisième établit que deux régions adjacentes qui satisfont chacune le prédicat $\mathcal{H}$ ne peuvent pas être fusionnées. Dans des images ambiguës, une solution analytique pour la segmentation est difficile à trouver tandis qu'une approche perceptuelle psychophysique deviendrait plus adaptée [Cheng 01].

La segmentation d'images a été un sujet de recherche actif pendant les trois dernières décennies. De nombreux algorithmes ont été développés pour des images monochromatiques. Cependant, le problème de la segmentation des images couleur, qui offrent beaucoup plus d'informations sur les objets de la scène, a fait l'objet d'une attention moindre de la part de la communauté scientifique [Comaniciu 02]. De nos jours, nous disposons de puissants systèmes de calcul qui nous permettent de traiter les images couleur texturées sans que le temps d'exécution soit un empêchement.

Plusieurs techniques de segmentation monochromatique peuvent être adaptées sans complication à la segmentation d'images couleur, en travaillant indépendamment sur les composantes chromatiques (R, G, B) ou sur les composantes d'une autre représentation de la couleur (voir Annexe § A). Selon les propriétés exploitées pour délimiter ou regrouper les pixels en régions, les méthodes de segmentation sont classifiées dans l'une des catégories suivantes : les approches basées sur la notion de région, les techniques qui se servent de l'information des contours de la scène et, enfin les méthodes qui travaillent par classification directe des pixels.

Étant donné que les techniques de segmentation sont très variées, mélangeant parfois plusieurs attributs [Derin 87, Altunbasak 98] (couleur, texture, mouvement, forme etc.), nous nous contenterons de présenter un état de l'art général de la segmentation couleur.

3.2.2 Méthodes de segmentation couleur

La segmentation couleur prend en compte la nature multi-spectrale de la lumière et son effet sur l'impression visuelle des objets présents dans la scène, pour extraire depuis une image, une information pertinente et utile pour d'autres applications, sous la forme des régions de couleur homogène [Luo 97].

La segmentation d'une image couleur, peut se faire à partir d'une quelconque représentation de la couleur : donc, la première étape de toute méthode de segmentation couleur, consistera à transformer l'image de l'espace *RGB* dans l'espace choisi. Nous décrivons les principales représentations de la couleur proposées dans la littérature, en Annexe § A.

Le choix de l'espace de couleur pour la segmentation d'une image, n'est pas facile et il est parfois lié au type d'application et à des facteurs très particuliers. Ainsi, la segmentation qui implique souvent l'utilisation d'un critère d'homogénéité dans un espace quelconque de couleur

a du mal à obtenir la séparation correcte des objets bien texturés, ombragés ou simplement en présence de reflets. En bref,

- les espaces linéaires ont l'inconvénient d'avoir des composantes chromatiques corrélées qui les rendent plus sensibles aux changements d'illumination.
- les approches non-linéaires sont algorithmiquement complexes et présentent des problèmes de singularités sur certaines régions du gammut.

Plusieurs approches de segmentation couleur sont basées sur des méthodes monochromatiques adaptées pour manipuler des pixels codés dans le système de couleur choisi. Parmi celles-ci on trouve les approches suivantes : la classification non supervisée ou *clustering* sur un espace d'attributs, le seuillage d'histogrammes, la croissance de régions, les méthodes fondées sur la détection de contours, les algorithmes fondés sur la logique floue, ceux qui utilisent des réseaux de neurones... Certaines approches appliquent la segmentation dite monochromatique sur chaque canal de manière indépendante pour ensuite fusionner les résultats partiels et obtenir un résultat global [Pal 93]. Par contre, les méthodes qui traitent simultanément les attributs couleur doivent gérer une quantité de données importante et souffrent de complexité algorithmique élevée. Les sections suivantes présentent de façon brève quelques unes de ces techniques.

3.2.2.1 Techniques de seuillage d'histogrammes

Le seuillage est certainement la technique la plus utilisée ; sur une image N&B, elle est fondée sur la supposition que les objets à segmenter et le fond de l'image ont des distributions[18] en niveaux de gris différentes [de Albuquerque 04]. Sous cette hypothèse, les histogrammes contiennent plusieurs pics ou modes représentant les objets. Il est donc possible d'obtenir les seuils de séparation sur l'histogramme qui permettront de dégager les objets dans l'image. La segmentation est menée à bien en affectant une même étiquette à l'ensemble des pixels entre deux seuils préalablement obtenus [Sahoo 04, Yan 03]. Dans l'application du seuillage sur des images couleur, des seuils sont recherchés indépendamment sur chaque attribut de l'espace couleur choisi ; la principale limitation repose sur le fait que les *clusters* obtenus seront des boîtes -rectangles en 2D, parallélépipèdes rectangles en 3D... (voir figure 3.2)- orientées selon les axes de l'espace des attributs. Ceci se répercute directement dans la qualité de la segmentation.

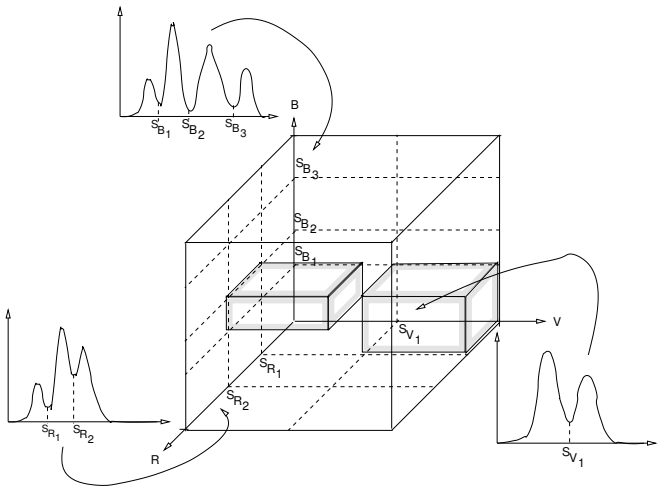


FIG. 3.2 – *Cluster* rectangulaire issu d'une technique de seuillage

[18] mot utilisé indistinctement pour décrire les histogrammes

Les méthodes de segmentation par seuillage peuvent être divisées en deux groupes : les méthodes globales et les méthodes locales. Dans les techniques globales, l'opération de seuillage utilise l'image entière $\mathcal{F}_I$ pour obtenir un histogramme, considéré comme la fonction de densité de probabilité d'un attribut pour les pixels de l'image. Les techniques locales sont basées sur la partition de l'image en plusieurs sous-images et l'obtention d'un seuil pour chacune des sous-images. Évidemment, les approches globales sont plus faciles à implémenter et demandent moins de ressources de calcul [Baradez 04].

Divers critères peuvent être utilisés pour détecter les seuils de segmentation à partir des histogrammes construits pour chaque attribut, sur l'image ou la sous-image à segmenter. Le but est de trouver un ensemble de seuils indépendamment pour chaque attribut. Par exemple, certaines approches utilisent la fonction d'entropie [de Albuquerque 04] comme principal critère [Yan 03] pour estimer le seuil.

La méthode d'Otsu effectue d'abord une analyse statistique directement sur les histogrammes (la variance intra-classe et la variance extra-classe) pour définir une fonction à maximiser qui permettrait d'estimer le seuil. C'est une technique développée initialement pour faire la classification des pixels en deux classes [Otsu 79] qui donne toujours un bon résultat pour des distributions bimodales. Dans le cas de distributions multimodales la méthode ne trouve que des résultats approximatifs.

Une méthode alternative basée dans le même esprit, sur la classification des pixels a été développée par Kittler [Kittler 86], inspiré par les travaux d'Otsu. Cette technique suppose que les observations (histogrammes) proviennent d'un mélange de distributions de type gaussien. Kittler a utilisé la théorie de l'information (relation de *Kullback-Leibler*) pour estimer les seuils à partir de la minimisation d'une fonction constituée par les paramètres de ces gaussiennes. Ces paramètres (μ, σ) sont calculés directement à partir des histogrammes et la relation de *Kullback-Leibler* est définie par $J = \sum_{i=1}^{n} P(i) log(\frac{P(i)}{f(i)})$, où $P(i)$ représente l'histogramme normalisé, n est le nombre de niveaux et $f(i)$ est le mélange de gaussiennes. Pour deux gaussiennes, la fonction clef $f(i)$ prend la forme suivante :

$$f(i) = \frac{q_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{i-\mu_1}{\sigma_1}\right)^2} + \frac{q_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{1}{2}\left(\frac{i-\mu_2}{\sigma_2}\right)^2} \tag{3.1}$$

Kittler utilise les mêmes relations statistiques que Otsu pour estimer le seuil t :
- les probabilités cumulées $q_1 = \sum_{i=1}^{t} P(i)$ et $q_2 = \sum_{i=t}^{n} P(i)$,
- les valeurs moyennes $\mu_1 = \sum_{i=1}^{t} iP(i)/q_1$, $\mu_2 = \sum_{i=t}^{n} iP(i)/q_2$
- et enfin les variances des gaussiennes, $\sigma_1 = \sum_{i=1}^{t}(i-\mu_1)^2 P(i)/q_1$, $\sigma_2 = \sum_{i=t}^{n}(i-\mu_2)^2 P(i)/q_2$ dans cette définition de $f(i)$.

Avec ces paramètres, l'équation de *Kullback-Leibler* est simplifiée et minimisée suivant t pour obtenir le seuil optimal sous ce critère.

Pal [Pal 91] a proposé une méthode similaire à celle de Kittler, utilisant à cette occasion une distribution de type *Poisson* $p(x) = \frac{\mu^x e^{-u}}{x!}$ pour représenter une fonction de probabilité sur les histogrammes.

Nous avons vérifié expérimentalement que la technique de Kittler donne des seuils plus précis en les comparant avec ceux obtenus par les techniques d'Otsu et de Pal. Liyuan Li a étendu la méthode unidimensionnelle d'Otsu au cas d'histogrammes bidimensionnels [Li 97], ce qui améliore considérablement la segmentation mais en augmentant le temps de calcul.

Par contre, le seuillage en utilisant directement des histogrammes 3D couleur n'est pas une tâche triviale car le temps de calcul pour trouver les *clusters* à partir de ces histogrammes 3D, deviendrait prohibitif. Des stratégies pour surmonter ces difficultés utilisent des arbres binaires en

projetant l'espace 3D couleur (par exemple, l'espace *rgb* normalisé), dans un espace de dimension réduite.

3.2.2.2 Méthodes de *clustering*

Les méthodes de *clustering* sont considérées comme une généralisation des techniques de seuillage dans l'espace 3D : en ce cas, l'espace des attributs est segmenté en zones qui ont une forme polyédrique quelconque. Une manière simple de segmenter les images consiste à exécuter une analyse sur l'agglomération des pixels couleur (*clusters*). Les pixels appartenant au même objet doivent être proches dans l'espace de couleur, et cet ensemble de pixels peut être modélisé statistiquement comme une distribution normale ; cette définition probabiliste permet d'établir des règles de séparation entre classes [Wang 03]. La *séparation* de l'espace est ensuite projetée vers l'image pour aboutir à la segmentation [Celenk 90].

L'algorithme *K-means* [Duda 98] est la plus populaire de ces méthodes de *clustering* ; mais la détermination automatique du nombre des *clusters* K (validation) reste un problème important, surtout pour une exploitation non supervisée. Plusieurs variantes ont ajouté de l'information supplémentaire : des opérations morphologiques, des informations spatiales [Matas 95] ou même des approches floues qui rendent le *clustering* plus robuste (*Fuzzy K-means*). Zoller [Zoller 02] a proposé une approche fondée sur des distributions paramétriques pour mener à bien le processus de *clustering* en utilisant la couleur et la texture. Il a validé son approche sur des scènes naturelles.

Un algorithme non paramétrique élégant a été mis au point par D.Comaniciu [Comaniciu 02] ; il est fondé sur l'estimation des gradients de densité sur la distribution des attributs. Il utilise un ancien concept issu de la reconnaissance de formes, appelé « Mean Shift » délimitant la forme des *clusters* dans un espace d'attributs multidimensionnel. Cette méthodologie est de plus en plus utilisée dans le cadre de la segmentation couleur, mais aussi pour la détection et le suivi temporel d'objets dans des séquences d'images. Le *Mean Shift* est une procédure itérative de recherche de maximum local dans un espace $\mathbb{R}^d$ servant à mesurer la similarité entre les distributions de couleurs d'un modèle de l'objet et celles de certaines régions de l'image.

Vu le succès de cette méthode, nous l'avons implémentée et évaluée. Nous avons détecté certaines limitations de cette technique (une légère sous-segmentation et un temps de calcul élevé) en l'appliquant sur nos scènes naturelles d'extérieur. Or, nous préférons travailler avec des images plutôt sur-segmentées car ce défaut peut être corrigé facilement en prenant en compte d'autres attributs (texture, forme, ...) tandis qu'une sous-segmentation est plus difficile à corriger avec notre architecture.

3.2.2.3 Techniques basées sur les régions

Ces approches fonctionnent en utilisant les propriétés intrinsèques des régions (homogénéité et connexité). Elles sont très utilisées en segmentation d'images car elles prennent en compte la couleur et l'information spatiale en même temps. Ce groupe inclut :

- les méthodes de croissance de régions : des régions élémentaires ou « germes » (*region seeds*) croissent, les unes en compétition avec les autres.
- et les méthodes par division et/ou fusion (*Split and Merge*) [Orteu 89].

Les méthodes de croissance de régions utilisent des *germes* de régions, puis intègrent dans ces régions élémentaires, les pixels de leur voisinage qui satisfont un critère d'homogénéité. Le processus est répété jusqu'à ce que tous les pixels de l'image soient classifiés. Évidemment, une limitation de ces méthodes est la détermination automatique des éléments dits *germes* et l'ordre dans lequel les pixels et régions sont examinés. Certaines approches utilisent un seuillage

d'histogrammes pour déterminer ces régions *germes*, en fonction de la dynamique des couleurs sur l'image [Avina-Cervantes 02].

L'algorithme de Deng détermine d'abord un nombre limité de classes dans l'image, en utilisant un prétraitement de *quantization*[19] [Deng 99] et il définit à cet effet un critère de « bonne » segmentation [Deng 01]. Deng utilise ce critère à des échelles différentes sur des fenêtres locales pour obtenir ce qu'il appelle les *J-images*. Dans ces images on trouve l'information des contours possibles inter-régions, ainsi que les centres des régions. La segmentation est réalisée par une croissance de régions où les régions *germes* sont obtenues à partir des vallées des *J-images* ; les erreurs possibles de sur-segmentation sont corrigées par une opération de fusion. Cette technique fonctionne correctement même pour des images texturées mais est trop coûteuse en temps de calcul.

Une méthode basée sur des divisions successives des régions a été proposée par Ohlander [Ohlander 78] pour la segmentation des images couleur. Ohta [Ohta 80] s'est servi de cette méthode pour réaliser ses expériences pour l'obtention de l'espace I_1, I_2, I_3 par transformations de Karhunen-Loève successives. La méthode de division ou découpage, part de l'image entière et la divise si un critère n'est pas satisfait. Le processus est ré-appliqué récursivement sur chaque division jusqu'à ce que le critère soit satisfait pour toutes les sous-parties de l'image.

Les algorithmes qui mélangent simultanément fusions et divisions successives sont appelés techniques de *split and merge*. Elles partent généralement d'une division de l'image selon un *quadtree*, avec une certaine taille pour les régions initiales. L'étape de division découpe les régions initiales non homogènes, tandis que l'étape de fusion part de chaque région initiale et les associe avec leur régions voisines si un critère est satisfait. Les deux processus sont ré-appliqués récursivement.

La plupart de ces approches profitent d'une structure de données appelée *graphe d'adjacence de régions* (RAG) pour la gestion de la segmentation. Saber [Saber 96] combine la couleur et l'extraction des contours pour développer une méthode *split and merge* robuste dans la délimitation des régions.

3.2.2.4 Techniques par détection de contours

Nous savons que la détection de contours est essentielle dans la perception des objets par l'être humain. Le concept de détection de contours a donc été exploité intensivement, surtout pour la segmentation d'images monochromatiques. De la même manière, sur des images couleur, certains algorithmes essaient de localiser des changements abrupts entre les points d'un voisinage [Gevers 02] par des opérateurs du type gradient ou laplacien. Les gradients peuvent être définis de deux manières différentes, soit avec un gradient couleur unique (opérateur de Di Senzo), soit avec une combinaison des N images obtenus par un gradient (opérateurs de Sobel, Deriche, Canny...) appliqué indépendamment sur chacune des composantes chromatiques. Le principal défaut de ces méthodes émerge au moment de traiter des images acquises sur des scènes très ombragées. Ceci peut être atténué en utilisant une composante chromatique stable en définissant par exemple un gradient ou un laplacien dépendant uniquement de la teinte dans l'espace *HSI*.

Notons qu'une image ne peut pas être segmentée en utilisant uniquement l'information de contours. Celle-ci doit être combinée avec d'autres méthodes pour compléter la segmentation : les méthodes de fermeture de contours sont généralement heuristiques et peu robustes. En pratique les algorithmes fondés sur les contours, fonctionnent sans complications quand le contraste entre régions est bien établi.

[19] La *quantization* consiste à réduire l'information couleur d'une image, *i.e.*, réduction du gammut (*e.g.*, de 24 à 8 bits)

Nous pouvons placer dans cette catégorie les méthodes de segmentation par contours actifs (*Color Snakes*) [Marin-Hernandez 04], très utilisées pour le suivi temporel. Elles sont mises en oeuvre à partir de l'application d'un gradient couleur sur l'image courante [Avina-Cervantes 03a]. Les *snakes* classiques considèrent la déformation d'un contour initial qui évolue vers le contour réel de l'objet à détecter. Pour cela, les déformations sont obtenues en minimisant l'énergie globale et le minimum local correspond à la frontière de l'objet : si le contour initial est éloigné du contour de l'objet à segmenter, ces méthodes sont peu robustes.

3.2.2.5 Approches physiques

Il existe aussi des approches qui exploitent des modèles physiques [Kim 96] d'interaction entre la lumière et la matière, ce qui implique une connaissance profonde sur la consistance et les propriétés physiques des objets qui composent l'environnement [Novak 90]. Ainsi, Healey [Healey 92] utilise un modèle de réflexion monochromatique pour déduire une modélisation statistique paramétrique des couleurs associées aux régions de l'image appartenant àux objets colorés. Un graphe binaire est rempli avec l'information des contours sur chaque bande de couleur, en commençant par l'image globale. Celle-ci est ensuite subdivisée (récursivement) en quatre régions identiques jusqu'à ce que ces sous-régions soient dépourvues d'information de contours (*i.e.*, régions homogènes). Les données de ces sous-images sont ajustées via un modèle statistique, les paramètres obtenus permettant de les fusionner avec toutes les sous-images voisines ayant des attributs similaires.

Certains algorithmes sont issus du domaine de la morphologie mathématique, comme la méthode de segmentation par ligne de partage des eaux (*Watershed*) [Brethes 04]. Elle considère l'image $\mathcal{F}_I$ comme une surface topographique définissant des bassins d'attraction et des lignes de partage des eaux avec une nomenclature des processus de débordement ou des inondations. Supposons que chaque cavité de la surface soit percée et que toute la surface soit plongée dans un lac à vitesse verticale constante. L'eau rentre à travers des trous inondant la surface. Au moment où l'inondation commence à remplir deux bassins différents et sur le point de les mélanger un barrage est placé pour éviter le mélange. L'union de tous les barrages définit les lignes de partage des eaux dans l'image qui nous conduiront à la segmentation.

Pour conclure sur ce panorama rapide des méthodes de segmentation d'images couleur, disons que les résultats obtenus par ces méthodes sont en général très variables. Puisqu'il n'y a pas de méthode universelle de segmentation d'images, il faut, en fonction du type d'images à traiter, faire appel à une technique ou à une autre ou même, fusionner les résultats obtenus par plusieurs méthodes appliquées simultanément. .

3.2.3 Notre méthode de segmentation couleur

Notre principal objectif est de proposer une méthode de segmentation rapide et efficace, applicable aux images acquises sur des scènes d'extérieur ou sur des objets naturels ; dans la mesure où cette méthode est le premier maillon dans notre approche d'analyse d'images couleur, il est essentiel qu'elle trouve les frontières entre les objets de classe différente ; par contre, nous pouvons tolérer une sur-segmentation qui pourra être corrigée ensuite. La technique choisie est hybride, au sens où elle combine la croissance de régions et le *clustering* par seuillage d'histogrammes. Nous avons raffiné l'algorithme proposé par R. Murrieta dans sa thèse de doctorat [Murrieta-Cid 98, Lasserre 96].

Cet algorithme est indépendant du point de départ de la segmentation car ce sont les techniques de seuillage qui détermineront les régions *germes* qui conduiront à une pré-classification

globale des pixels dans toute l'image $\mathcal{F}_I$. Les régions voisines de la même classe sont groupées à l'aide d'un critère global et un graphe d'adjacence de régions (en 8-connexité). Pour éviter une trop forte sur-segmentation, les régions de taille inférieure à un seuil donné seront fusionnées à la région voisine la plus proche dans l'espace de couleur.

Les étapes principales de l'algorithme sont décrites ci-après :

1. **Changement d'espace de couleur**. Dans nos expériences, l'espace couleur d'Ohta est celui qui a donné le meilleur compromis entre qualité de segmentation et temps de calcul. Étant donné que certaines composantes dans les espaces de couleur (*cf.* Annexe § A) ont une dynamique de valeurs impropres à l'élaboration des histogrammes ($\mathcal{F}_I(x,y) \notin [0,255]$), une normalisation dynamique de chaque composante (*autoscaling*) peut être effectuée de deux manières :
   – en utilisant l'équation de ré-haussement des images obtenues pour chaque composante :

$$\mathcal{F}'_I(x,y) = \mathcal{A}\,\frac{\mathcal{F}_I(x,y) - \mathcal{I}_{min}}{\mathcal{I}_{max} - \mathcal{I}_{min}} + \mathcal{B} \tag{3.2}$$

   où $\mathcal{I}_{max}$ et $\mathcal{I}_{min}$ sont respectivement les valeurs maximale et minimale de l'image d'origine (avec éventuellement un *max* supérieur à 255 ou un *min* négatif) tandis que $\mathcal{A} = 255$ et $\mathcal{B} = 0$ sont respectivement les nouvelles valeurs maximale et minimale.
   – en redéfinissant l'espace de couleur. Par exemple, pour l'espace de Ohta (*cf.* équation A.7) nous utilisons une transformation qui produit des valeurs I_1, I_2, I_3 directement exploitables dans les histogrammes,

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & -1/2 \\ -1/4 & 1/2 & -1/4 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128.5 \\ 128.5 \end{pmatrix}, \tag{3.3}$$

   D'après nos expériences, la normalisation pour la mise à l'échelle de chaque composante entre 0 et 255 (équation 3.2), ajoute une opération supplémentaire par rapport au simple changement d'espace. Ainsi, les contours sont légèrement décalés sur chaque composante chromatique ce qui peut rendre la segmentation mauvaise dans certains cas. De plus, plusieurs valeurs sur les histogrammes disparaissent à cause des problèmes d'arrondi dans cette opération, ce qui perturbe aussi les techniques de seuillage. Nous utilisons donc l'espace de Ohta modifié (attributs non corrélés) qui ne requiert pas d'opérations additionnelles.

   Cet espace s'est avéré le plus approprié pour la majorité des scènes traitées. Pour certaines images où la saturation est plutôt faible, il est possible de renforcer les contours pour maintenir une bonne qualité de segmentation.

2. **Estimation des *clusters***. Ceux-ci sont obtenus à partir des différents seuils qui ont été estimés sur les histogrammes représentant les trois composantes couleur : ce sont donc des bôites rectangles. Les méthodes de seuillage que nous utilisons sont celles d'Otsu, Pal et plus particulièrement celle de Kittler (décrites en § 3.2.2.1). Notre implémentation nous permet d'estimer de trois à cinq classes pour chaque bande de couleur. Ceci va nous permettre de classifier chacun des pixels sur l'image en réduisant en même temps le nombre de couleurs.

3. **Pré-classification (étiquetage)**. Cette opération permet d'affecter chaque pixel de l'image couleur, à une des composantes connexes formées dans l'espace des attributs couleur, par les *clusters* rectangulaires (*cf.* figure 3.2). Cette classification dans un cluster, peut se faire à une résolution quelconque, éventuellement non uniforme en lignes et colonnes. Nous

pouvons ainsi classifier des régions élémentaires à plusieurs résolutions $\mathcal{R}_c = 1 \times 1, 2 \times 2, 3 \times 3, \ldots$ en utilisant à cet effet leurs moyennes colorimétriques :

$$m_{\mathcal{R}_c} = \frac{1}{n_{\mathcal{R}_c}} \sum_{\{x,y\} \in \mathcal{R}_c} \mathcal{F}_I(x, y) \tag{3.4}$$

4. **Croissance à partir des régions élémentaires optimales**. Cette opération fait la fusion de régions de la même classe en utilisant un graphe d'adjacence de régions par connexité à 8 voisins. Après cette phase, tous les pixels de l'image sont affectés à une région mère qui contient plusieurs informations comme l'étiquette de la classe et la moyenne colorimétrique de tous les pixels appartenant à cette région.
5. enfin, **Fusion (élimination) des petites régions** . Les régions de surface inférieure à un seuil donné sont fusionnées à la région connexe (région mère) la plus proche dans l'espace couleur, en utilisant la distance euclidienne $d(\mathcal{R}_{C_1}, \mathcal{R}_{C_2}) = \|\mathcal{R}_{C_1} - \mathcal{R}_{C_2}\|$, où $\mathcal{R}_{C_1}$ et $\mathcal{R}_{C_2}$ sont les vecteurs couleur associés aux deux régions à comparer. Lors de cette phase, nous utilisons un algorithme de suivi de contours (*cf.* § 4.3.1) qui nous permet d'obtenir rapidement des graphes de connexité inter-régions, ou simplement de trouver les régions (ou pixels) voisins d'une région donnée.

Dans ce processus, c'est paradoxalement la dernière étape, l'élimination des petites régions, qui est l'opération la plus coûteuse. Ceci a des conséquences sur les autres étapes ; par exemple, dans la phase de croissance des régions, on pourrait utiliser une connexité à 4 voisins pour limiter la complexité ; mais en fait, la 4-connexité rend globalement la segmentation beaucoup plus lente qu'une 8-connexité, car cela génère des petites régions qu'il faut ensuite filtrer.

En utilisant l'intensité comme paramètre de segmentation (I_1), en sus des attributs de chromaticité (I_2 et I_3), nous obtenons parfois des images sur-segmentées. Cet effet est du principalement à la présence de fortes variations d'intensité et à la complexité de l'environnement. Mais ce phénomène n'est pas gênant car on corrige cet inconvénient après une classification des régions, en fusionnant les régions connexes qui appartiennent à une même classe (voir la figure 3.14). Ainsi, la caractérisation des régions par la texture permet de corriger les images sur-segmentées.

3.3 Méthodes d'estimation de la texture

Bien que la texture soit visible de manière naturelle par le système visuel humain, en donner une définition est très complexe. Cette difficulté est mise en évidence par les diverses définitions de texture disponibles dans la littérature. Ainsi, actuellement, il n'existe aucune règle mathématique de caractère général qui soit capable de quantifier les différentes classes de texture.

Cependant, les propriétés intuitives les plus utilisées pour décrire la texture sont les suivantes :

– la texture est une propriété liée à une région support, qui doit être uniforme au sens sémantique. Elle n'est donc pas définie en un point, mais sur un ensemble de pixels dans un voisinage spatial de ce point. Par exemple, calculer la texture sur une région image acquise pour une partie sur une route uniforme, et pour une autre partie, sur un bas-côté herbeux, n'a pas de sens ; les régions support pour le calcul de la texture doivent donc être bien définies.
– la texture est le reflet de l'analyse d'une distribution spatiale de pixels en niveaux de gris dans la région support, *e.g.*, des histogrammes ou des matrices bidimensionnelles de co-occurrence,

– la texture dans une image peut être perçue et mesurée à différents niveaux de résolution,
– une région est dite texturée quand le nombre de primitives caractérisant l'objet dans la région est grand et quand aucune forme individuelle (significative) n'est présente.

De nombreux travaux ont été dédiés à ce sujet ce qui nous permet de disposer de différents opérateurs pour mesurer la texture : les méthodes structurelles et géométriques, les méthodes basées sur un modèle physique, les méthodes par filtrage et les méthodes statistiques. Toutes ces méthodes sont utilisées sur les images de luminance : nous n'avons pas analysé de travaux sur des attributs de texture couleur, qu'ils soient issus de calculs spécifiques, ou de combinaison de textures calculées sur chaque composante.

3.3.1 Approches structurelles et géométriques

Ces approches considèrent que les textures sont formées par des motifs ou des primitives réguliers qui se répètent périodiquement (ou presque) dans une région limitée de l'espace, par exemple le carrelage sur le sol, un mur de briques. Il est clair que cette définition n'est pas générale, car dans la nature beaucoup d'objets sont aléatoirement texturés (par exemple l'écorce des arbres, les feuillages d'une haie, la terre labourée d'un champ, le revêtement dégradé sur une route...) : cette méthode n'est donc valable que pour des applications spécifiques. Elle n'est pas applicable dans notre contexte de textures naturelles.

3.3.1.1 Modèles géométriques

La méthode d'analyse dépend habituellement des propriétés géométriques des éléments de texture. Une fois que les éléments de texture sont identifiés dans l'image, il y a deux approches à suivre. En premier lieu, on calcule les propriétés statistiques des éléments extraits et on utilise ces derniers directement comme des attributs de texture. L'autre approche essaie d'extraire la primitive et la règle de placement qui décrit la texture. Cette dernière approche peut inclure des méthodes géométriques ou syntactiques pour analyser la texture.

La tessellation par Voronoi a éte proposée comme un modèle pour définir un ensemble de « voisinages » caractérisant une région. Pour extraire les marques (*tokens*) de texture, on procède comme suit : (1) on applique à l'image $\mathcal{F}_I$ un filtrage par différences de gaussiennes (laplacien $\nabla^2\mathcal{F}_I$), (2) on choisit les pixels qui se trouvent sur un maximum local d'intensité dans l'image filtrée (binarisation), (3) on réalise une analyse de connectivité de composantes sur l'image binaire en utilisant les huit plus proches voisins. Chaque composante connexe définit une primitive de texture.

La tessellation par Voronoi est construite en utilisant ces primitives. Des attributs sont extraits à partir de chaque cellule et les primitives ayant les mêmes attributs sont groupées pour former des régions uniformes de texture. Ainsi, les moments calculés sur les polygones de Voronoi sont utilisés comme des attributs pour représenter la distribution spatiale (texture) et la forme des *tokens* dans l'image. Les moments d'ordre supérieur m_{pq} caractérisant la texture sur une région R sont :

$$m_{pq} = \iint_R (x - x_o)^p (y - y_o)^q \, dxdy \tag{3.5}$$

où (x_o, y_o) est la position de la primitive ou *token*.

3.3.1.2 Modèles structurels

Les modèles structurels, supposent que les textures sont composées de primitives élémentaires qui sont organisées suivant une règle de placement définie. Ces algorithmes ne sont bien adaptés que pour des textures très régulières (par exemple, mur de brique). Certaines approches considèrent l'image comme un ensemble de primitives de texture arrangées suivant une règle de placement. En ce cas, l'analyse structurale de texture est donc composée de deux étapes : extraction des primitives de texture et recherche d'une inférence servant de règle de placement inter-primitives.

La primitive peut être un pixel mais en général c'est un ensemble de pixels formant un motif. La règle de placement est définie par un graphe grammatical. Une texture est alors considérée comme une chaîne dans un langage qui suit des règles grammaticales formées par les primitives de texture. Un avantage de cette méthode est qu'elle peut être employée pour la génération de texture aussi bien que pour l'analyse de texture [Chen 98].

3.3.2 Méthodes fondées sur des modèles physiques ou *templates*

Ces méthodes exploitent un modèle d'un motif-image appelé *template*, qui sert à caractériser la texture ou à la synthétiser. Les paramètres de ces modèles doivent capturer les qualités essentielles qui caractérisent la texture perçue. Les champs aléatoires de Markov (MRF's) et les approches par *fractales* (autosimilarité à des échelles différentes) sont deux exemples typiques de ce type de représentation de la texture.

3.3.2.1 Champs aléatoires de Markov

Les modèles markoviens, tout particulièrement les champs de Markov et les champs de Gibbs, sont devenus des outils indispensables pour traduire des informations complexes en une loi de probabilité qui les représente. En effet, la modélisation markovienne permet de construire des modèles qui peuvent prendre en compte les discontinuités dans un signal ou sur une image [Geman 84]. Cet outil a une importance toute particulière en traitement d'images où l'on peut construire des modèles composites (contours, intensité) pour les images et les utiliser lors de la segmentation, restauration ou reconstruction d'images [Kim 96].

Ces approches sont capables de capturer des informations spatiales de type contextuel au moins localement. Elles considèrent que l'intensité en un pixel spécifique est une fonction directe de l'intensité mesurée sur les pixels voisins [Derin 87].

Dans les *MRF*, l'image est représentée par un treillis de taille $M \times N$, noté $L = \{(i,j) \mid 1 \leq i \leq M\,, 1 \leq j \leq N\}$. $I(i,j)$ est une variable aléatoire représentant le niveau de gris de chaque pixel (i,j) sur le treillis L. Pour simplifier l'indexation du treillis, il est intéressant d'utiliser la notation I_t où $t = (i-1)N + j$. L'ensemble de toutes les étiquettes dans L est représenté par $\omega = \{(x_1, x_2, \ldots, x_{MN}) \mid x_t \in A, \forall t\}$ où A est l'ensemble de toutes les variables aléatoires représentées par I_t.

Un *MRF* discret est un champ aléatoire ayant une densité de probabilité qui respecte les propriétés de positivité, de markovianité et d'homogénéité. L'ensemble des voisins t peut être de premier ordre (4-voisinage) ou de deuxième ordre (8-voisinage). La probabilité conditionnelle utilise alors un ensemble de *cliques* (site simple, double, triple, quadruple) basées sur les voisinages antérieurement définis [Cross 83]. La modélisation d'une texture consiste à obtenir les paramètres de toutes les cliques du *MRF*, en utilisant par exemple un modèle de *GIBBS* pour définir sa probabilité conditionnelle correspondante.

Pour un système du type (L, η) le MRF est défini par les caractéristiques des probabilités locales $P(X_{ij} = x_{ij}|X_{kl}, (k,l) \neq (i,j))$. La densité de probabilité X_{ij} est conditionnée seulement par le système de voisinage η centré sur (i,j). Une clique c pour (L, η) est définie par un sous-ensemble de L tel que : (1) c soit un singleton ou, (2) $(i,j) \in c, (k,l) \in c \Rightarrow (i,j) \in \eta_{kl}$.

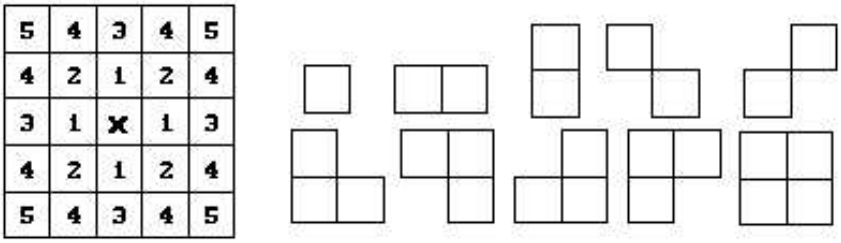


FIG. 3.3 – a) À gauche le système de voisinage hiérarchique, b) à droite les cliques pour le système de voisinage du deuxième ordre.

Pour un ensemble (L, η) un champ aléatoire X est un champ de Gibbs, si sa distribution de probabilité a la forme suivante :

$$P(X = x) = \frac{1}{Z} e^{[-\beta U(x)]} \qquad \forall x \in \omega \tag{3.6}$$

où $U(x) = \sum_{c \in Q} V_c(x)$ est la fonction d'énergie qui est généralement exprimée en termes de cliques, $V_c(x)$ est le potentiel associé à la clique c, Q est la collection de toutes les cliques de L et $Z = \sum_x -\beta U(x)$ est une constante de normalisation appelée la fonction de partition.

X_{ij} est une variable aléatoire qui prend des valeurs dans un ensemble $Q = \{q_1, q_2, q_3, \ldots, q_M\}$ et qui assigne un paramètre à chaque type de clique. La fonction potentiel de la clique est définie par

$$y = \begin{cases} -\beta_k & \text{si tous les } X_{ij} \text{ et } c_k \text{ sont égaux} \\ \beta_k & \text{sinon .} \end{cases} \tag{3.7}$$

Il sera donc possible de modéliser la texture en utilisant l'énergie locale ou globale donnée par les interactions pixels (voisinages), ce qui est représenté globalement par une loi de probabilité.

3.3.3 Méthodes fondées sur un filtrage

Ces méthodes exploitent le fait que le système visuel humain réalise des analyses fréquentielles de l'image. Or, la texture est spécialement adaptée à ce type d'analyse.

3.3.3.1 Filtres spatiaux

Les filtres spatiaux permettent d'obtenir simplement de l'information de texture perdue dans l'image. Les premières approches mesuraient la densité des contours par unité de surface. Ainsi, les contours sont obtenus généralement en utilisant les opérateurs de Roberts (M_1,M_2) et le laplacien (L) :

$$M_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.8}$$

J.Malik [Malik 90] a proposé un filtrage spatial pour modéliser la perception pré-attentive de texture dans le système visuel humain. Cette modèlisation se compose de trois étapes : (i) la convolution de l'image avec un banc de filtres symétriques, suivi d'une rectification à demi onde, (ii) la suppression des réponses parasites dans une région déterminée, et (iii) la détection des frontières entre les textures différentes.

D'autres types de filtres spatiaux, sont basés sur des moments d'ordre supérieur qui sont mesurés sur tous les pixels à l'intérieur d'une région R,

$$m_{pq} = \sum_{(x,y)\in R} x^p y^q \, I(x,y) \tag{3.9}$$

L'exploitation des moments est équivalent à l'utilisation d'un ensemble de masques pour le filtrage spatial. Les attributs de texture sont donc les images obtenues du filtrage en utilisant les moments d'ordre supérieur. Cette analyse est facilement adaptée à la segmentation d'images.

3.3.3.2 Domaine de Fourier

L'analyse fréquentielle d'images texturées est naturellement réalisée dans le domaine de Fourier car elle permet d'obtenir les composantes de fréquence et de phase d'une zone texturée. L'integrale de Fourier est définie par :

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} \, dx \tag{3.10}$$

Les filtres de Fourier permettent de réaliser des analyses de texture à plusieurs résolutions en utilisant des filtres sélectifs par fréquence et par orientation. Ce type de représentation de texture est capable de segmenter et de classifier des images naturelles ou synthétiques.

3.3.3.3 Filtres de Gabor et transformation en ondelettes

La transformée de Fourier réalise une analyse globale fréquentielle des signaux mais certaines applications ont besoin d'une analyse focalisée dans un intervalle spatial donné. Une manière pour ce faire, consiste à utiliser la transformée de Fourier pondérée par une fenêtre $w(x)$. Cette opération est représentée, pour un signal unidimensionnel par :

$$F_W(u,\xi) = \int_{-\infty}^{\infty} f(x) w(x-\xi) \, e^{-j2\pi ux} \, dx \tag{3.11}$$

La fenêtre $w(x)$ détermine la qualité (résolution) de la bande passante dans ce filtre. Si $w(x)$ est de type gaussien, cette transformation est appelée la transformée ou filtre de *Gabor* [Jain 97]. En 1946, Gabor a introduit la transformée de Fourier à fenêtre (Short-Time Fourier Transform, STFT) afin de mesurer « les variations fréquentielles » des sons. La résolution en temps et fréquence est donnée par l'inégalité d'incertitude de Heisenberg, $\Delta t \Delta u \geq 1/4\pi$.

La résolution temps-fréquence est fixée sur tout le plan temps-fréquence dès que la fenêtre utilisée dans la transformée de Fourier est choisie. Pour pallier ces inconvénients, il est avantageux d'utiliser une fenêtre de dimension variable en fonction des variations de fréquence. La transformation utilisant cette fenêtre adaptative qui permet d'analyser des composantes transitoires de durée différente, est connue comme la transformée par ondelettes, elle est donnée par :

$$W_{f,a}(u,\xi) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t) h^* \left(\frac{t-\xi}{a} \right) \, dt \tag{3.12}$$

où $h(t) = w(t)e^{-j2\pi ut}$ est la réponse impulsionnelle (fenêtre $w(t)$ modulée en fréquence), a représente le paramètre de dilatation de l'ondelette. La résolution temps-fréquence pour les ondelettes est aussi liée au principe d'incertitude de Heisenberg. Les ondelettes et les transformées de Fourier par fenêtre ont une énergie bien localisée dans le temps, tandis que leur transformée de Fourier est essentiellement concentrée sur un intervalle de fréquences relativement étroit.

Comme une transformée de Fourier par fenêtre, une transformée en ondelettes permet de mesurer l'évolution en temps de composantes fréquentielles. Pour cela, il faut une ondelette analytique complexe, afin de séparer la phase et l'amplitude. Par contre, les ondelettes réelles sont souvent utilisées pour la détection des transitions abruptes d'un signal ou des contours dans une image. Elles permettent aussi d'analyser les structures locales d'un signal avec un zoom qui réduit progressivement le paramètre d'échelle. En appliquant ces transformations, il est donc possible de réaliser des analyses de texture à plusieurs résolutions, avec des contraintes imposées selon l'application.

En pratique, les filtres de Gabor bidimensionnels les plus utilisés sont basés sur des fonctions sinusoïdales d'onde plane (à une certaine fréquence et orientation) modulées par une fonction gaussienne. Ces filtres sélectifs sont représentés par :

$$f(x,y) = cos(2\pi u_o x + \phi)\, exp\left\{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right\} \tag{3.13}$$

où u_o et ϕ sont respectivement la fréquence et la phase de l'onde sinusoïdale. Les paramètres σ_x^2 et σ_y^2 sont respectivement les dimensions de l'enveloppe gaussienne dans les directions x et y.

Un filtre de Gabor peut être conçu en choisissant une fenêtre W, dont les dimensions se calculent à partir des fréquences centrales du filtre. Les attributs de texture sont déterminés comme suit :

- utiliser un banc de filtres à plusieurs fréquences et orientations pour obtenir les images filtrées $r_i(x,y)$,
- faire subir aux images filtrées $r_i(x,y)$, une opération non-linéaire $\Psi(t)$ de type sigmoïde $tanh(\alpha t)$,
- estimer les attributs de texture sur chaque pixel, en utilisant l'écart absolu moyen sur les valeurs des images filtrées centrées à l'intérieur d'une fenêtre W de dimension $M \times M$.

Les images filtrées ont donc une moyenne nulle et le $i^{ième}$ attribut de texture se calcule par :

$$e_i(x,y) = \frac{1}{M^2} \sum_{(a,b)\in W} \mid \Psi(r_i(a,b)) \mid \tag{3.14}$$

où M est déterminé en utilisant la fréquence centrale du filtre. Ces attributs de texture ont régulièrement été utilisés en segmentation et classification dans la littérature donnant des résultats satisfaisants [Campbell 97], au prix de temps de calcul très importants.

3.3.4 Modèles statistiques

Ces méthodes n'exploitent pas la notion de motif. Elles utilisent des attributs statistiques définis sur des pixels et leur voisinage comme primitives élémentaires de texture [Hauta-Kasari 96]. Ces textures peuvent avoir un aspect désordonné mais sont cependant considérées comme homogènes ce qui rend ce type de modèles très adaptés aux textures naturelles. Les modèles statistiques de texture les plus représentatifs, exploitent les populaires matrices de co-occurrence, les coefficients d'auto-corrélation et la représentation que nous avons choisie, les histogrammes de sommes et différences.

Nous décrivons ci-après ces deux modèles. Nous décrivons ensuite, la méthode originale proposée par R.Sanchez, appelée *Coordinated Cluster Representation* (CCR) ; nous collaborons avec R.Sanchez pour évaluer cet attribut sur notre base d'images, notamment pour l'exploiter dans la phase de segmentation.

3.3.4.1 Matrices de co-occurrence

Les matrices de co-occurrence ont été l'une des premières méthodes disponibles dans la littérature pour coder la texture dans une image ; plusieurs travaux ont été menés pour les généraliser [Haralick 93]. Cette représentation estime les propriétés de l'image par des statistiques issues de la distribution spatiale des pixels. [Haralick 79] a proposé l'ensemble d'attributs de texture les plus utilisés pour mesurer la signature d'une région. Ces attributs sont calculés directement à partir des matrices de co-occurrence. La matrice de co-occurrence de taille $N \times N$, notée $M_{\boldsymbol{d}}$, est définie comme suit :

$$\boldsymbol{M_d}(i,j) = \#\{((r,s),(t,v)) :\ \mathcal{F}_I(r,s) = i,\ \mathcal{F}_I(t,v) = j\} \tag{3.15}$$

où $\mathcal{F}_I$ est l'image d'intensité ayant N niveaux de gris et $\boldsymbol{d} = (dx, dy)$ est le vecteur de déplacement associé à la distance $((r,s),(t,v))$. En effet, $M_{\boldsymbol{d}}(i,j)$ représente le nombre d'occurrences des paires de niveaux de gris (i,j) pour deux pixels séparés spatialement d'une distance $\boldsymbol{d}$.

L'implantation de ces matrices présente quelques inconvénients : il n'existe pas de critère qui nous permette d'estimer le paramètre de distance $\boldsymbol{d} = (dx, dy)$, et plusieurs valeurs sont possibles, donnant probablement des résultats très différents. Du point de vue pratique, il est prohibitif de calculer plusieurs matrices pour des déplacements différents et de fusionner les résultats. En plus, parmi ces descripteurs de texture, une sélection des attributs les plus discriminants doit être effectuée, car tous ne peuvent être exploités.

3.3.4.2 Attributs d'autocorrélation

Les attributs d'autocorrélation considérent la nature répétitive de la texture sur l'image ; la fonction d'autocorrélation peut être exploitée pour mesurer la régularité (homogénéité) ainsi que la finesse de la texture sur l'image. Elle est définie comme suit :

$$\rho(x,y) = \frac{\sum_{u=0}^{N}\sum_{v=0}^{N}\mathcal{F}_I(u,v)\mathcal{F}_I(u+x,v+y)}{\sum_{u=0}^{N}\sum_{v=0}^{N}|\mathcal{F}_I|^2(u,v)}\,. \tag{3.16}$$

Cette signature est directement associée à l'échelle de la texture à mesurer via (x,y). De plus, la fonction d'autocorrélation est bien liée à la densité spectrale de puissance de la transformée de Fourier sur l'image (*cf.* théorème de Wiener-Kintchine). Il est donc possible d'obtenir des attributs spectraux en discrétisant le domaine de fréquences, *i.e.*, l'énergie dans chaque sous-région est considérée comme un attribut de texture.

3.3.4.3 Histogrammes de sommes et différences

Les histogrammes sont considérés comme une signature pour une image ou une région. M. Unser a donc proposé une méthode basée sur des histogrammes monodimensionnels pour estimer la texture sur les régions d'une image [Unser 86]. Il est évident que la complexité algorithmique de cette approche est moindre que celle des matrices de co-occurrence. Ainsi, cette approche a besoin d'un espace de mémoire réduit et c'est une excellente approximation des mesures proposées par Haralick.

Pour une région donnée d'une image $\mathcal{F}_I(x,y) \in [0,255]$, les histogrammes de sommes et de différences sont définis de la façon suivante :

$$h_s(i) = \#\{(\mathcal{F}_I(x,y) + \mathcal{F}_I(x+\delta x, y+\delta y) = i\} \tag{3.17}$$
$$h_d(j) = \#\{|\ \mathcal{F}_I(x,y) - \mathcal{F}_I(x+\delta x, y+\delta y)\ | = j\} \tag{3.18}$$

où $i \in [0,510]$ et $j \in [0,255]$. Les images de somme et de différence peuvent être établies pour tous les pixels (x,y) de l'image d'entrée $\mathcal{F}_I$,

$$I_s(x,y) = \mathcal{F}_I(x,y) + \mathcal{F}_I(x+\delta x, y+\delta y) \tag{3.19}$$
$$I_d(x,y) = |\ \mathcal{F}_I(x,y) - \mathcal{F}_I(x+\delta x, y+\delta y)\ | \tag{3.20}$$

Il est donc possible de caractériser un échantillon spécifique de texture par une collection d'histogrammes de sommes et différences estimés pour différents déplacements relatifs δx et δy. En outre, les histogrammes de sommes et différences normalisés peuvent être calculés pour des régions choisies dans l'image, de sorte que :

$$H_s(i) = \frac{\#(I_s(x,y) = i)}{m}, \quad H_s(i) \in [0,1] \tag{3.21}$$
$$H_d(j) = \frac{\#(I_d(x,y) = j)}{m}, \quad H_d(j) \in [0,1] \tag{3.22}$$

où m est le nombre de points qui appartiennent à la région. Ces histogrammes normalisés peuvent être interprétés comme suit : $\widehat{\boldsymbol{P}}_{s(i)} = H_s(i)$ est la probabilité que la somme des pixels $I(x,y)$ et $I(x+\delta x, y+\delta y)$ ait la valeur i ; $\widehat{\boldsymbol{P}}_{d(j)} = H_d(j)$ est la probabilité que la différence absolue des pixels $I(x,y)$ et $I(x+\delta x, y+\delta y)$ ait la valeur j.

En utilisant ces probabilités, on dispose d'une caractérisation probabiliste de l'organisation spatiale des pixels dans une région basée sur une analyse de voisinage. Les attributs les plus représentatifs dérivés de ces distributions sont résumés dans la table 3.1. La valeur moyenne μ qui apparaît dans toutes ces relations, est obtenue par la relation suivante :

$$\mu = \frac{1}{2} \sum_i i \cdot \widehat{\boldsymbol{P}}_{s(i)}\,. \tag{3.23}$$

Notons que la plupart de ces expressions sont très similaires aux attributs proposés par Haralick pour les matrices de co-occurrence mais elles sont obtenues avec une complexité algorithmique moindre. Notons que le calcul de ces attributs se fait pour des décalages δx,δy déterminés ; notre stratégie pour choisir ces décalages va dépendre de la phase pour laquelle ce calcul de texture est effectué.

Étant donné que les ressources de calcul et de mémoire sont limitées sur un système embarqué sur un robot, nous avons adopté cette approche pour la caractérisation des régions d'une image acquise sur une scène naturelle, par des attributs de texture. C'est une méthode rapide qui permet d'extraire des descripteurs de texture avec une quantité limitée de mémoire : nous évaluons aussi la représentation CCR qui pourrait aussi se révéler intéressante vu sa simplicité de mise en oeuvre.

3.3.4.4 Représentation de la texture par « Clusters Coordonnés »

La représentation d'une image par CCR (Coordinated Clusters Representation), est proposée par R.Sanchez [Sánchez-Yáñez 03b], comme une alternative rapide pour caractériser la texture dans une image de niveaux de gris [Sánchez-Yáñez 03a].

Attributs	**Formules de texture**
Énergie	$\sum_i \widehat{\boldsymbol{P}}^2_{s(i)} \cdot \sum_j \widehat{\boldsymbol{P}}^2_{d(j)}$
Corrélation	$\frac{1}{2}(\sum_i (i-2\mu)^2 \cdot \widehat{\boldsymbol{P}}_{s(i)} - \sum_j j^2 \cdot \widehat{\boldsymbol{P}}_{d(j)})$
Écart-type	$\frac{1}{2}(\sum_i (i-2\mu)^2 \cdot \widehat{\boldsymbol{P}}_{s(i)} + \sum_j j^2 \cdot \widehat{\boldsymbol{P}}_{d(j)})$
Entropie	$-\sum_i \widehat{\boldsymbol{P}}_{s(i)} \log \widehat{\boldsymbol{P}}_{s(i)} - \sum_j \widehat{\boldsymbol{P}}_{d(j)} \log \widehat{\boldsymbol{P}}_{d(j)}$
Contraste	$\sum_j j^2 \cdot \widehat{\boldsymbol{P}}_{d(j)}$
Homogénéité	$\sum_j \frac{1}{1+j^2} \cdot \widehat{\boldsymbol{P}}_{d(j)}$
Nuance	$\sum_i (i-2\mu)^3 \cdot \widehat{\boldsymbol{P}}_{s(i)}$
Proéminence	$\sum_i (i-2\mu)^4 \cdot \widehat{\boldsymbol{P}}_{s(i)}$
Pmax	$\max\{\widehat{\boldsymbol{P}}_{s(i)}\}$

TAB. 3.1 – Attributs de texture calculés à partir des histogrammes de sommes et différences

Pour calculer la CCR sur une image primaire de niveaux de gris Im de L lignes et M colonnes, il est nécessaire de choisir une fenêtre rectangulaire de dimension $W = I \times J$ pixels ($I < L$ et $J < M$). La fenêtre W permet de faire une exploration séquentielle de l'image, par balayage par pas de un pixel. Une sous-image de taille W est centrée en chaque pixel et est d'abord binarisée après recherche d'un seuil optimal ; cette sous-fenêtre est ensuite associée à une variable $b \in \left[0, 2^W - 1\right]$ qui est la valeur décimale correspondant à la chaîne de bits obtenue en concaténant, ligne après ligne, les valeurs binaires des pixels de la fenêtre. Le mot (nombre) binaire b est considéré comme un code pour la configuration donnée par la fenêtre ; il codifie les structures répétitives sur la fenêtre.

La représentation par « Clusters Coordonnés » d'une image, est l'histogramme sur les occurrences des structures répétitives obtenues en chaque pixel par l'opération décrite ci-dessus. Étant donné qu'avant codage, une binarisation est effectuée localement pour une fenêtre de dimension $W = I \times J$, le nombre de tous les états possibles est 2^W. Cette valeur détermine la « taille » d'un histogramme primaire, taille qui peut être réduite significativement si sont écartés tous les états ayant des occurrences nulles lors de l'exploration de l'image, ou en considérant seulement un nombre limité de *baquets* pour construire cet histogramme. Donc, la représentation par *Clusters Coordonnés* consiste à obtenir l'histogramme $H_{(I,J)}(b)$ avec les occurrences des structures répétitives données par fenêtrage sur l'image d'origine. Ici, les sous-indices (I, J) indiquent la taille de la fenêtre d'exploration.

Une distribution de probabilité d'occurrences est alors obtenue, en normalisant l'histogramme

$H_{(I,J)}(b)$:

$$F_{(I,J)}(b) = \frac{1}{A} H_{(I,J)}(b) \tag{3.24}$$

où $A = (L-I+1) \times (M-J+1)$ représente le nombre de pixels pour lesquels le code b est calculé sur le voisinage de taille (I, J). En effet, les bords de l'image d'origine ne sont pas considérés : si $I = 2i+1$ et $J = 2j+1$ sont impairs, l'image est balayée de la ligne i à la ligne $L-i-1$, soit un balayage de $L-I+1$ lignes ; de même, pour chaque ligne, seul $M-J+1$ colonnes sont balayés.

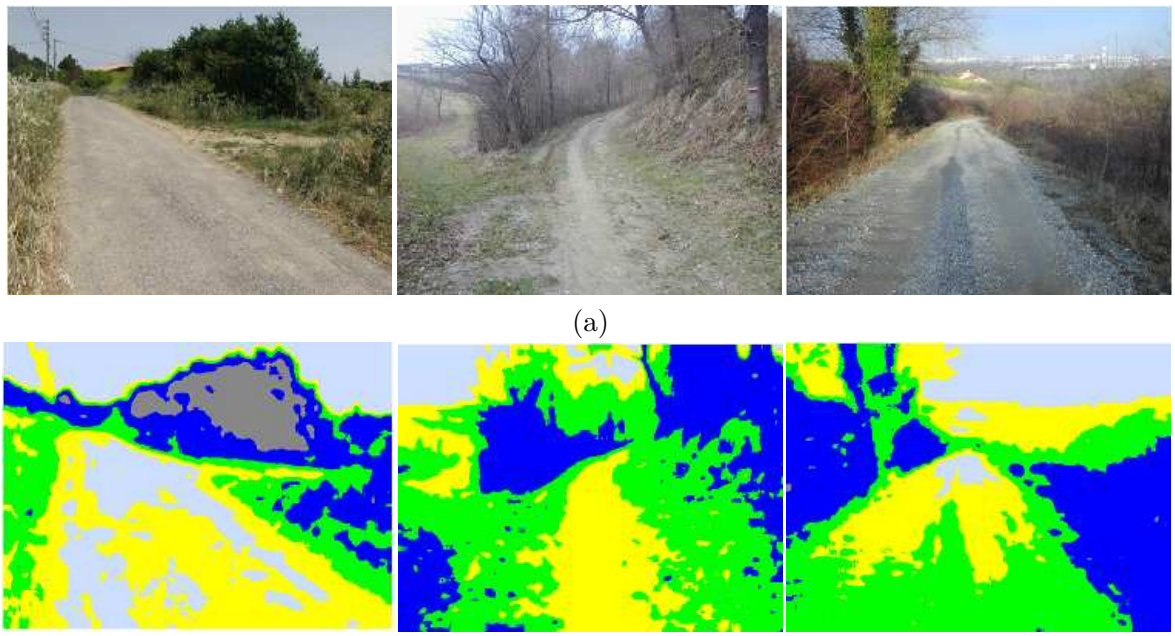
Une des propriétés fondamentales [Sánchez-Yáñez 03b] de la CCR établit la relation entre $H_{(I,J)}(b)$ et les moments du $n^{ième}$ ordre de la corrélation spatiale d'une image binaire. L'histogramme $H_{(I,J)}(b)$ contient donc toute l'information à propos des moments de corrélation de n points dans l'image primaire en niveaux de gris Im, si et seulement si les vecteurs de séparation entre les n pixels sont contenus dans la fenêtre d'exploration. Cela signifie que la fonction de distribution $F_{(I,J)}(b)$ fournit l'information suffisante sur la probabilité conjointe des n points.

Il est connu que les fonctions de densité de probabilité de second ordre et d'ordre supérieur, fournissent des informations structurelles à propos de la répartition des niveaux des gris dans une image. Réciproquement, il existe une correspondance structurelle entre une image en niveaux de gris Im et l'image binaire produite par un seuillage à partir de Im. Lorsque l'image binaire fournit suffisamment d'informations structurelles sur l'image primaire en niveaux de gris à analyser, la CCR de l'image binaire est adéquate pour la reconnaissance et la classification des textures dans l'image en niveaux de gris [Sánchez-Yáñez 03b, Sánchez-Yáñez 03a].

La représentation de la texture par la CCR est conceptuellement et mathématiquement différente de celles données par (1) la méthode proposée par Wang et He [Wang 90], (2) la transformée Census proposée par Zabih et Woodfill [Zabih 94] et (3) la *LBP* (pour *Linear Binary Pattern*) explorée par Ojala et al. [Ojala 96]. Dans ces trois méthodes une codification binaire est aussi utilisée pour la description des structures répétitives.

Nous illustrons sur trois images acquises sur des chemins, l'exploitation de ce concept de CCR dans le cadre de la segmentation d'images. La figure 3.4 montre les résultats de segmentation en appliquant la Représentation des « Clusters Coordonnés », générée ici uniquement sur l'image d'intensité pour modéliser la texture. Nous constatons que la seule utilisation de l'intensité pour la segmentation ne suffit pas sur ce type d'images, *e.g.*, problèmes de sous-segmentation et de détection peu fidèle de la frontière entre régions. Nous avons alors ajouté une composante additionnelle pour améliorer la segmentation, soit un attribut de couleur de l'espace L^*ab, soit un autre attribut de texture issus des histogrammes de sommes et différences (*cf.* figure 3.5). Nous avons trouvé que la combinaison entre le CCR évalué uniquement sur l'intensité et la composante chromatique b (CIE-Lab) donnent les meilleures résultats. Ainsi, le développement d'une technique hybride (texture en utilisant CCR et couleur en utilisant uniquement la composante b) rapide pour des applications en robotique a donné des résultats préliminaires satisfaisants ; néanmoins, cette méthode requiert plus d'expérimentation et de validation.

Nous avons présenté un panorama global des techniques les plus utilisées pour quantifier la texture ; de ce fait, nous avons pu justifier le choix de la méthode utilisée dans notre approche pour décrire la texture. Dans la section suivante, nous utilisons les attributs de couleur, de texture et de contexte pour caractériser les régions issues de l'image segmentée.



(a)

(b)

FIG. 3.4 – Segmentation d'images par texture en utilisant la CCR. (a) Images d'origine, (b) images segmentées (sans analyse de connexité entre régions)

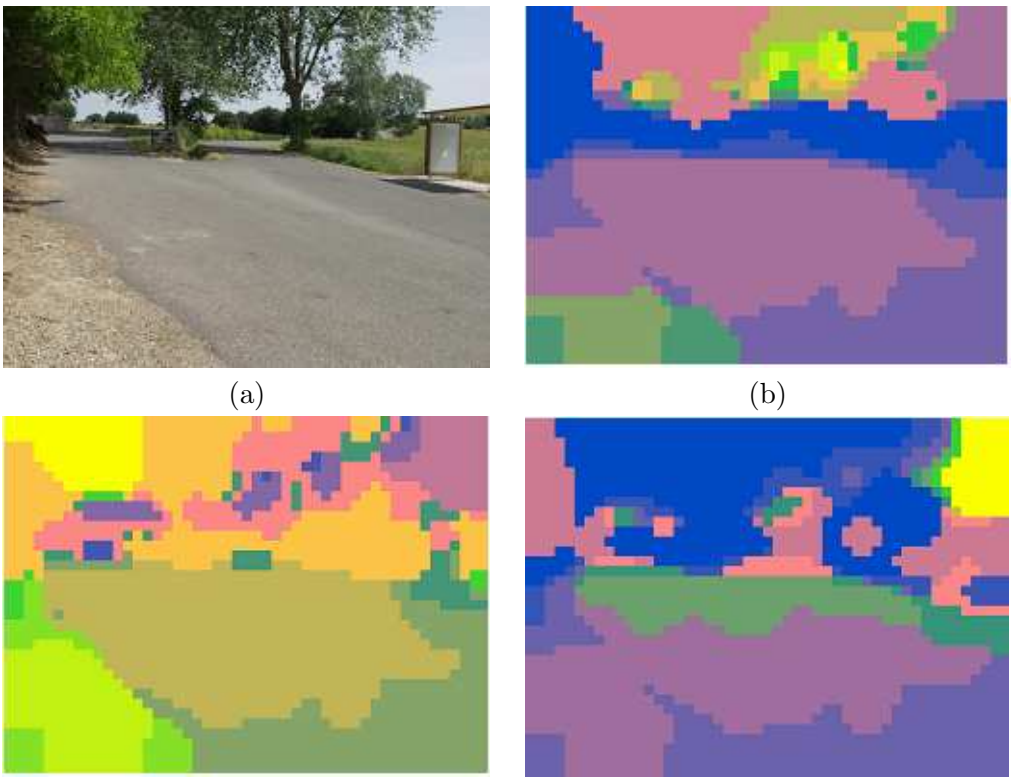
3.4 Caractérisation des régions

Dans notre approche générale, nous avons opté pour l'utilisation des informations de couleur afin de segmenter l'image en régions ; les informations de texture sont prises en compte après la segmentation, pour caractériser et classifier les régions. Pourquoi ce choix ? Nous aurions pu en effet, soit commencer la segmentation sur les attributs de texture, soit segmenter dans un espace d'attributs de plus grande dimension, combinant couleur et texture. Deux justifications :

- comme vu ci-dessus, la texture est définie obligatoirement sur une région ; dans notre cas, elle est estimée à partir des histogrammes de sommes et différences calculés sur les régions issues de la segmentation couleur. Les régions supports du calcul de la texture, ne sont pas définies arbitrairement, mais sont le résultat de la segmentation couleur.
- Généralement, les contours définis uniquement par des discontinuités de texture sont très imprécis ; dans notre approche, les contours des régions sont définis uniquement sur la base de la couleur.
- une méthode de *clustering* sur un espace hybride couleur-texture, ne pourrait pas prendre en compte un grand nombre d'attributs, afin de limiter les temps de calcul. Quels attributs faudrait-il privilégier parmi les 12 possibles (3 de couleur, 9 de texture) ?

Signalons toutefois, que dans la période actuelle, nous comparons plusieurs exploitations possibles des attributs pour la segmentation initiale : notre choix (segmentation uniquement sur la couleur) ou celui défendu par R.Sanchez-Yanez (segmentation sur un attribut chromatique et un ou deux attributs de texture issus de la représentation CCR).

Pour caractériser chacune des régions issues de la segmentation couleur, nous utilisons des attributs de couleur et de texture, afin de mieux les différencier. Ces attributs peuvent être renforcés avec des informations contextuelles comme la position et la forme.



(a) (b) (c) (d)

FIG. 3.5 – Segmentation sur des attributs hybrides. (a) Image d'origine, (b) Segmentation sur le CCR et le contraste, (c) Segmentation sur le CCR et la composante b de l'espace CIE-Lab, et (c) Segmentation par CCR et l'homogénéité

La sélection pertinente des attributs de bas niveau est importante pour mieux caractériser les régions. L'ensemble des attributs doit être déterminé en fonction de son pouvoir discriminant. Il est aussi important de considérer l'invariance des attributs à plusieurs facteurs comme la rotation, la translation, les conditions d'illumination, les changements de point de vue, les occultations, la distance entre le capteur et la scène (échelle)...

Les défaillances d'invariance pourront être atténuées en réalisant un apprentissage qui prend en compte les différentes conditions de l'environnement. Néanmoins, cette solution n'est pas la panacée : pour éviter une explosion combinatoire soit au moment de l'apprentissage des attributs de chaque classe, soit au moment de l'identification des régions, il faut aussi limiter la taille de la base d'images.

Les attributs de couleur sont définis ci-après, ceux de texture sont présentés en § 3.4.2 ; les attributs contextuels seront discutés en § 3.6.4. Pour justifier la complémentarité entre attributs de couleur et de texture, nous présentons en § 3.4.3 une analyse de la variation des attributs en fonction de la profondeur.

3.4.1 Calcul des attributs de couleur

Toutes les régions de l'image segmentée sont caractérisées par leur couleur en utilisant la moyenne statistique de leur trois composantes colorimétriques $\mathcal{F}_{C_i}$. Ces moyennes sont calculées par l'équation suivante

$$m_{\mathcal{R}_s}(c_i) = \frac{1}{n_{\mathcal{R}_s}} \sum_{(x,y)\in\mathcal{R}_s} \mathcal{F}_{C_i}(x,y) \quad \forall c_i \in \{c_1, c_2, c_3\}\,, \tag{3.25}$$

où $\mathcal{R}_s$ représente chacune des régions segmentées, $n_{\mathcal{R}_s}$ le nombre de points de cette région, c_i sont les trois plans couleur. Pour des raisons de performance, nous réutilisons les composantes couleur générées lors de l'étape de segmentation (*i.e.*, les composantes I_1, I_2 et I_3) ce qui permet de diminuer le temps de calcul.

3.4.2 Calcul des attributs de texture

Dans la section § 3.3, nous avons présenté les principales méthodes de représentation de la texture, ce qui va nous permettre de bien justifier la méthode de texture choisie pour notre application. Étant donné que notre système est embarqué sur un robot muni d'une capacité limitée de stockage et de manipulation de données, la méthode que nous avons choisie est celle de M.Unser, *i.e.*, des attributs calculés à patir des histogrammes de sommes et différences.

Ces histogrammes nous permettent de disposer d'un vecteur caractéristique représentatif pour chaque texture dans la scène. Ainsi, les attributs de texture sont calculés en choisissant les déplacements relatifs δx et δy selon les critères suivants :

- pour limiter la taille du vecteur caractéristique d'une région ainsi que le temps de calcul dans la *phase classification*, nous calculons les sommes et différences pour un seul déplacement relatif (*e.g.*, $\delta x = 1$ et $\delta y = 1$). En effet, nous avons choisi ces valeurs après avoir vérifié que les attributs de texture calculés pour d'autres déplacements relatifs dans le voisinage proche de chaque pixel$\{(\delta x, \delta y)\} \in \{-1, 0, 1\}/\{(0,0)\}$ donnent des grandeurs équivalentes,
- par contre, lors de la *phase d'apprentissage*, la base de connaissances est construite prenant en compte les huit directions élémentaires pour les déplacements relatifs, $\{(\delta x, \delta y)\} \in \{-1, 0, 1\}/\{(0,0)\}$. Ainsi, la valeur caractéristique de texture sur chaque région est obtenue utilisant la moyenne statistique de l'ensemble de textures rapportées par ces huit déplacements. L'idée consiste à éviter de privilégier une seule direction dans l'élaboration de la base d'apprentissage.

Une fois déterminés les paramètres impliqués dans le calcul de la texture, il reste à identifier les meilleurs attributs de texture parmi ceux du tableau 3.1. Nous voulons ne sélectionner que les attributs de texture (et couleur) les plus discriminants, afin de réduire la dimension de l'espace des attributs ; pour cela, nous avons réalisé une analyse en composantes principales (ACP). Cette technique nous a aussi permis de développer des stratégies pour mieux classifier les régions issues de la segmentation. De plus, une analyse en composantes indépendantes (ACI) a été aussi implémentée dans le but d'améliorer l'identification des objets sur la scène. Ces deux techniques seront décrites en § 3.5 comme pré-traitements sur la base d'apprentissage. Ainsi, elles nous ont permis de choisir les sept attributs suivants, parmi ceux du tableau 3.1) pour la caractérisation de régions par la texture : l'énergie, la corrélation, le contraste, l'entropie, l'homogénéité, la nuance (*cluster shade*) et la proéminence.

Mais quel est le degré d'invariance de ces attributs de texture en fonction de la résolution des images ? Cette question est d'autant plus pertinente que, dans une image acquise sur un terrain depuis un robot terrestre, des zones identiques de terrain (chemin, herbe, champ labouré) peuvent être perçues avec des effets perspective très différents selon qu'elles sont près ou loin du robot.

3.4.3 Variation des attributs de texture en fonction de la profondeur

L'étude classique de la texture ne prend généralement pas en compte la variation naturelle de l'échelle de texture provoquée par des effets de profondeur. Beaucoup de travaux sur la texture sont académiques, avec des résultats présentés sur les images de Brodatz sur lesquelles il n'existe pas de problèmes de profondeur. Ces problèmes n'existent pas non plus dans les applications réelles pour lesquelles la texture est souvent exploitée : imagerie aérienne (reconnaissance des cultures), inspection sur des objets plans (plaque de marbre...), reconnaissance de la nature d'un terrain depuis un véhicule, via une caméra avec l'axe optique orienté vers le terrain (travaux du C.Debain [Khadraoui 98] au CEMAGREF). Mais dans notre application, un champ labouré, le gazon sur un terrain plat, un long chemin, un long mur de briques, auront des mesures de texture différentes avec des variations importantes de l'angle de vue.

D'après nos expériences, il n'est pas rare de confondre un tronc d'arbre ou un champ labouré lointain avec un chemin ; ces types de phénomènes sont provoqués tout simplement parce que le pouvoir discriminant des descripteurs est négativement affecté par la profondeur. Il va de soi que l'être humain manifeste une forte capacité pour la reconnaissance des objets lointains, en exploitant principalement la couleur et l'organisation conceptuelle de la scène lorsque l'information de texture est réduite.

Nous avons étudié la variation des nos attributs de texture (issus des histogrammes de sommes et différences) en fonction de la profondeur. La figure 3.6 présente les courbes normalisées de la variation des paramètres retenus de texture et de couleur pour la classe chemin. L'énergie et l'homogénéité ont tendance à croître avec la profondeur ; l'entropie, le contraste, la corrélation et la proéminence ont un comportement décroissant également bien défini. Par contre, la nuance présente une haute sensibilité aux variations locales de texture ; les pics visualisés sur cet attribut sont sans doute liés à l'inhomogénéité dans l' échantillon de texture choisi pour cette étude.

Pour le terrain labouré (*cf.* figure 3.7), nous constatons la même tendance des attributs de texture en fonction de la profondeur que pour l'échantillon de chemin. De ce fait, comme une grande quantité de chemins de terre sont tracés a côté de champs labourés ayant la même constitution physique (couleur, ...) et étant différentiables facilement de près par des mesures de granularité ou d'homogénéité, il ne sera donc pas possible de classifier correctement ces deux classes de loin en utilisant uniquement la texture.

Les figures 3.8 et 3.9 présentent respectivement les courbes de variation de texture pour un échantillon de route et un échantillon d'herbe sur un terrain plat. Nous notons également que les observations effectuées sur les deux classes sont très similaires aux classes précédentes.

Nous considérons que ce type de comportement *régulier* sur les valeurs de texture en fonction de la profondeur est très important. Nous avons envisagé de corriger ou même de compenser des variations sur les attributs de texture dues aux paramètres tels que l'angle de vue ou la profondeur. Cela consisterait à développer des fonctions sensibles à la profondeur permettant de raffiner les attributs de texture ; ceci demanderait une estimation indirecte et rapide de la profondeur, soit en utilisant une *hypothèse de sol plat* (*cf.* § 5.3.4) pour les classes citées ci-dessus, soit sur une mesure approximative basée sur la position des objets dans l'image. Finalement, nous avons renoncé à cette compensation, lui préférant l'introduction d'attributs contextuels, présentés dans la section suivante.

En définitive, il n'est pas raisonnable de concevoir un système de reconnaissance d'objets en milieu naturel qui n'utilise que la texture ou que la couleur. En effet, les deux attributs sont complémentaires :

– dans des situations où les images ont des couleurs faiblement saturées (en hiver, en automne, dans les déserts, ...) l'importance de la texture est indéniable ;

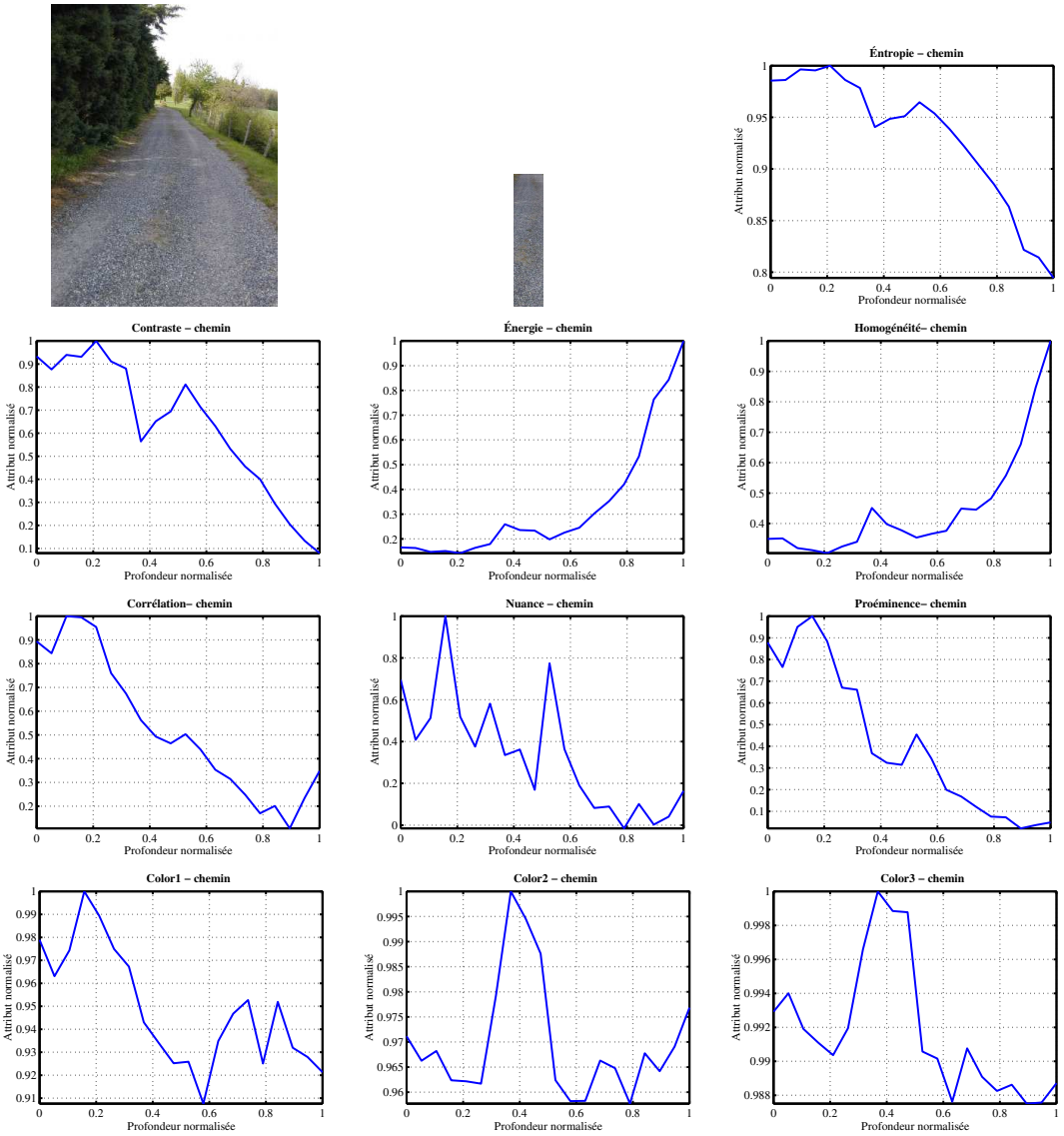


FIG. 3.6 – Variation des attributs de texture en profondeur pour un échantillon de chemin

– par contre, quand la texture perd son pouvoir discriminant sur des objets lointains, la couleur dévient la plus riche source d'information [Gevers 99].

Texture et couleur étant complémentaires, les régions segmentées sont caractérisées par un vecteur dans $\mathbb{R}^{12}$, qui comporte sept attributs issus des histogrammes de sommes et différences, trois attributs pour la couleur représentée dans l'espace de Ohta, et deux attributs contextuels de position qui seront rajoutés pour prendre en compte certaines connaissances sémantiques (par exemple, la probabilité d'avoir le chemin en bas de l'image est plus élevée) et rendre ainsi notre approche plus robuste (*cf.* § 3.6.4).

Dans les sections suivantes, nous présentons les méthodes qui exploitent ces vecteurs d'attributs, soit pour construire une base d'apprentissage, soit pour classifier les régions, pour parvenir enfin à la description de la scène, et à l'extraction de zones navigables.

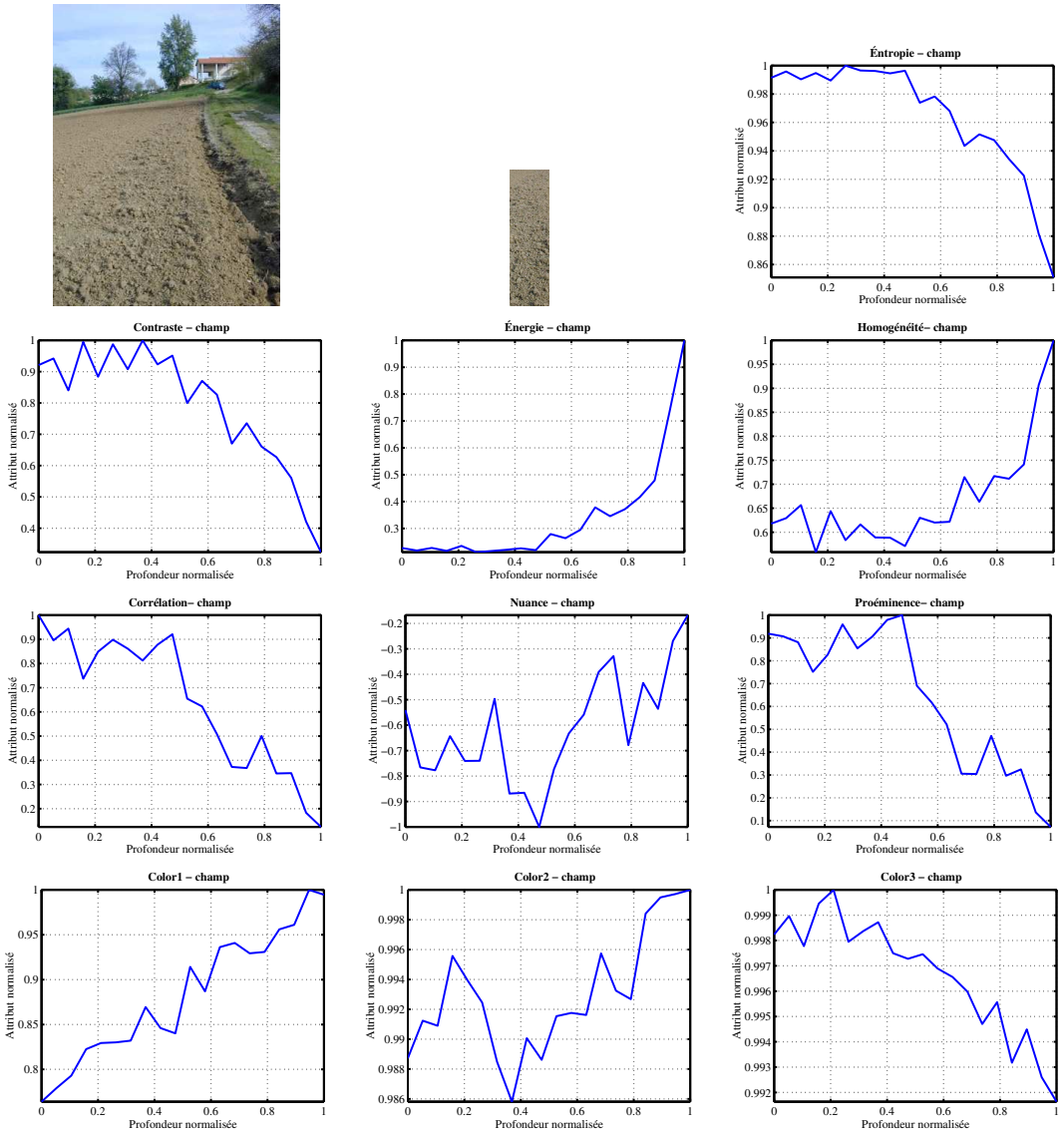


FIG. 3.7 – Variation des attributs de texture en profondeur pour un échantillon de champ labouré

3.5 Construction et analyse de la base d'apprentissage

Nous avons choisi l'espace des attributs dans lequel chaque région segmentée sera représentée. Notre objectif est d'étiquetter chaque région, en fonction de la nature de l'objet de la scène dont elle est l'image. Nous devons hors-ligne, apprendre les valeurs caractéristiques des attributs pour chaque classe des objets d'intérêt dans notre application ; comme la classification implémentée dans ces travaux, exploite des méthodes non paramétriques (SVM ou k-PPV), il s'agit en fait d'apprendre un grand nombre d'échantillons pour chaque classe.

Pour limiter la taille mémoire de cette base d'apprentissage, cette section est dédiée aux méthodes qui permettent de réduire la dimension globale des données apprises, soit en détectant les attributs les plus discriminants, soit en utilisant une nouvelle représentation des données par projection dans un autre espace. Ainsi, les méthodes de classification seront basées sur une base de données qui sera la mieux conditionnée possible.

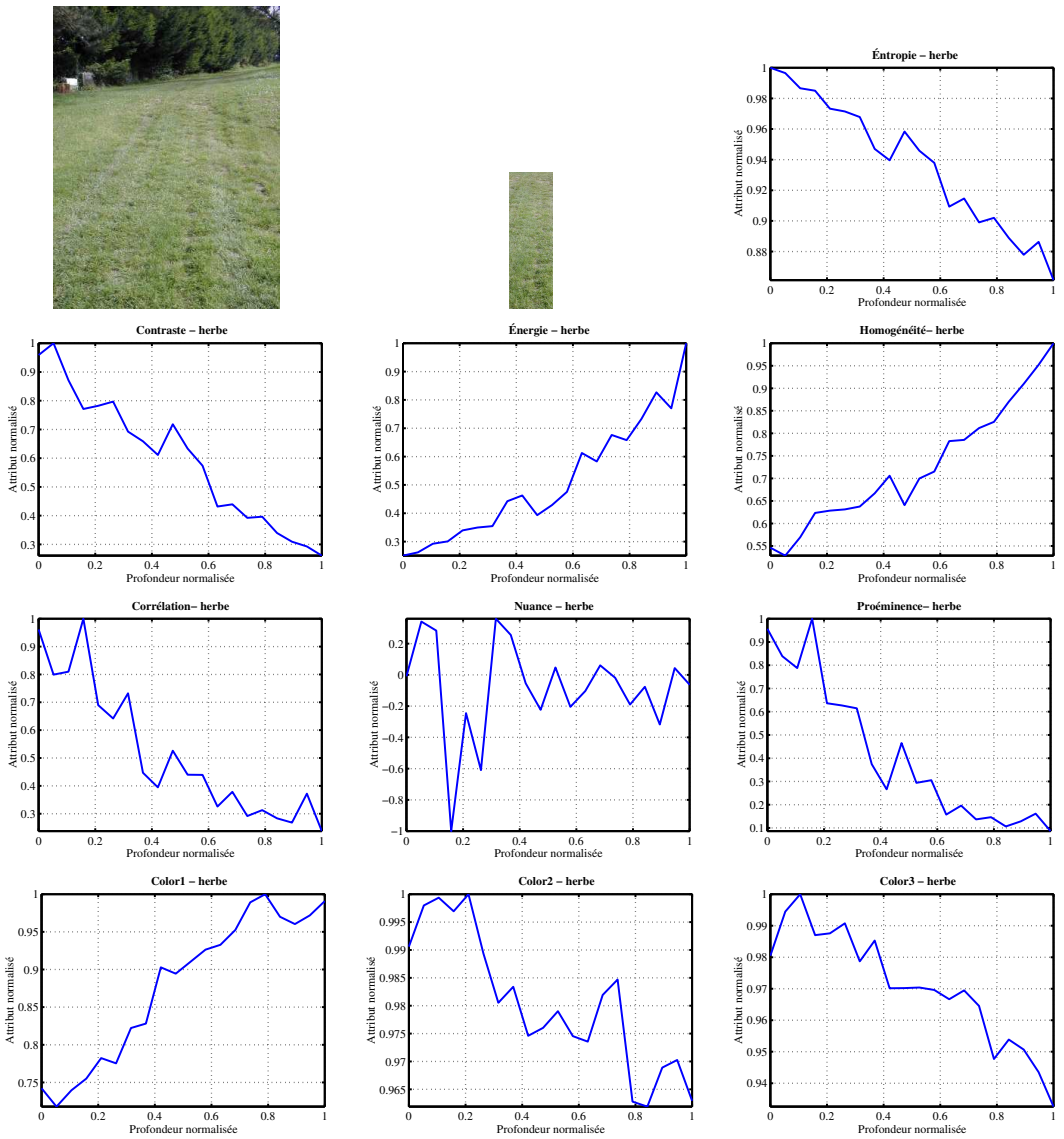


FIG. 3.8 – Variation des attributs de texture en profondeur pour un échantillon de terrain d'herbe

3.5.1 Apprentissage supervisé de la base de données

La base de données a été construite en mode supervisé ce qui a nécessité la détermination préalable des classes d'objet pertinentes pour l'application visée. Le nombre de classes peut être variable, pour s'ajuster aux variations de l'environnement. Un fichier contient initialement le nom des classes qui seront utilisées lors de la classification des régions, parmi lesquelles *CIEL, ARBRE, HERBE, CHAMP, CHEMIN, BOIS* et *EAUX*. Chaque classe est associée

- à une étiquette (un niveau unique, pour générer les images des régions étiquetées),
- et à un fichier portant le même nom de la classe, rempli des échantillons dans $\mathbb{R}^{12}$ des régions apprises de cette classe, sur un grand nombre d'images acquises en milieu naturel.

Le défi au moment de réaliser l'apprentissage consiste à maximiser les informations utiles, tout en minimisant la taille de la base de données (évitant le problème de sur-apprentissage), de temps d'accès et de calcul. Une analyse en composantes principales permet de déterminer le

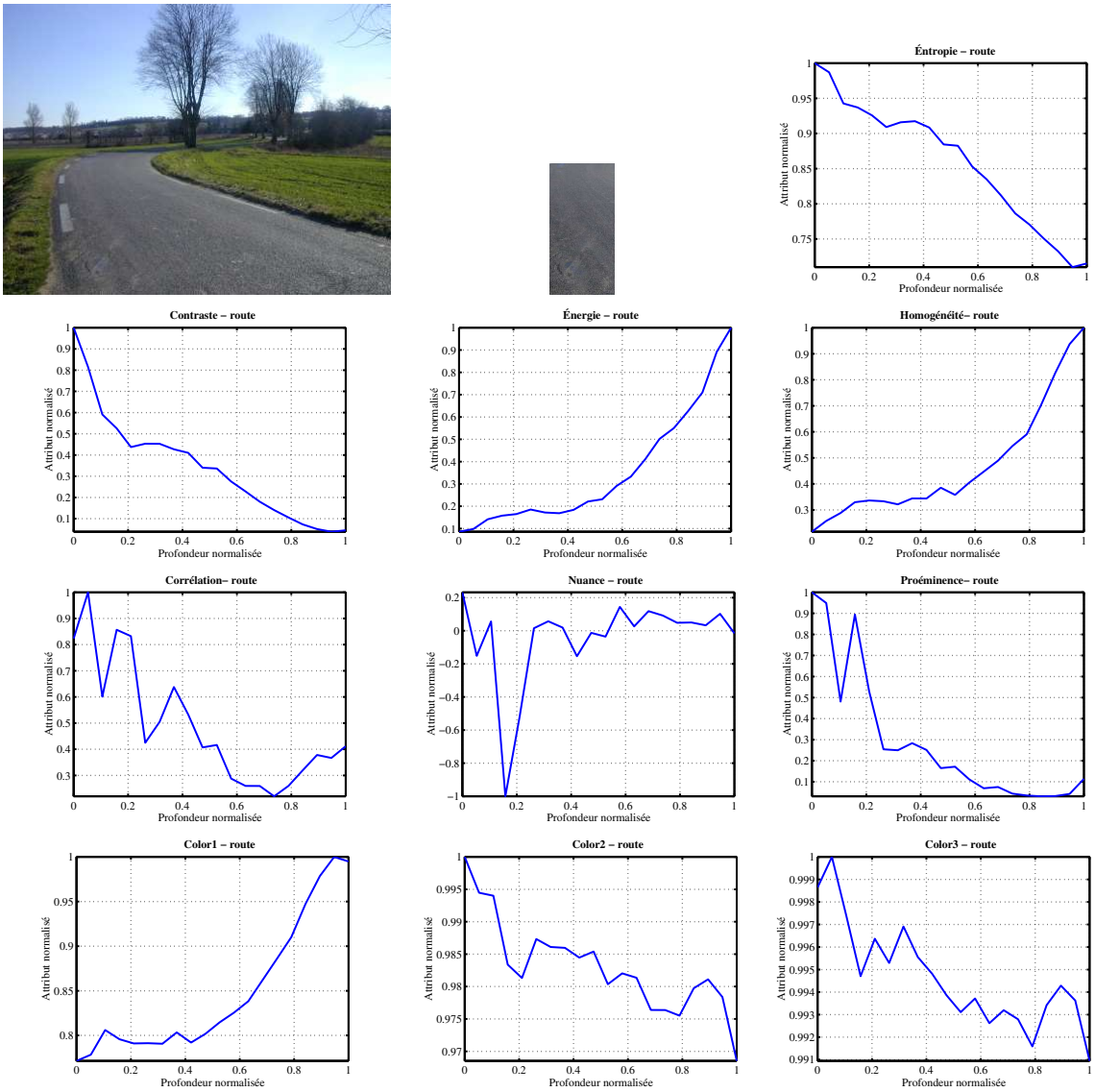


FIG. 3.9 – Variation des attributs de texture en profondeur pour un échantillon de route

nombre d'attributs les plus discriminants (information utile). Puis, des méthodes de réduction de données sont utilisées pour maintenir une dimension raisonnable (donc, diminuer le temps de classification) de la base de données et éliminer en même temps le bruit (*outliers*) généré lors de l'apprentissage.

3.5.1.1 Mise à jour en continu de la base de données

Nous avons utilisé un ensemble initial relativement limité d'images d'apprentissage (approximativement 40), que nous avons ensuite complété en fonction des résultats des tests, par des images sur des nouvelles entités mal classifiées, et apparaissant de façon récurrente. En effet, l'apprentissage est réalisé en continu depuis la constitution initiale de la base d'apprentissage et la définition des classes à identifier :

– ainsi, une classe non considérée dans le processus initial d'apprentissage, peut être rajoutée

ensuite, lorsque les objets correspondants sont perçus dans la scène. Typiquement, ces objets n'auront pas été reconnus (puisque non appris) ou auront été confondus avec une classe existante. C'est le superviseur qui doit décider de l'ajout d'une nouvelle classe (par exemple, nous avons rajouté la classe *EAUX* à une base existante déjà depuis plusieurs mois).
- de nouveaux échantillons peuvent être introduits pour une classe, si le superviseur constate qu'elle est mal identifiée dans certaines situations, hors d'atteinte du pouvoir de généralisation des classifieurs. La méthode de classification (*k-PPV*) peut indiquer au superviseur quels éléments dans les images de la base de données ont été « mal appris » requérant donc de les renforcer avec de nouveaux échantillons.

Il est important de préciser que lors de l'apprentissage, une image peut contribuer uniquement par quelques régions, jugées caractéristiques par le superviseur ; nous préférons utiliser une variété plus sélective d'échantillons provenant de plusieurs images pour éviter le sur-apprentissage.

L'apprentissage s'effectue surtout si des fausses détections augmentent (causées par changements de saison, d'illumination, *etc.*). Si ce phénomène est trop fréquent, la meilleure solution consiste à changer de base d'apprentissage, car il convient d'éviter les bases de données volumineuses qui peuvent ralentir la reconnaissance, (*cf.* § 3.5.1.2) ; signalons la technique de préclassification proposée dans [Murrieta-Cid 02], qui consiste à reconnaître d'abord un contexte global (été/hiver ou soleil/couvert ou urbain/campagne/désert . . .) afin de choisir la base d'apprentissage qui permettra d'analyser la scène courante.

3.5.1.2 Simplification de la base de données

Nous avons utilisé les approches suivantes pour maintenir la base de données à une taille raisonnable de N échantillons :

1. **une méthode de classification** (SVM en ce cas), classifie les éléments d'une même classe et seulement les éléments positivement reconnus, sont maintenus dans cette catégorie,
2. **un processus de *bootstrapping*** nous permet de réduire efficacement le nombre d'échantillons de la manière suivante :
   - choisir un échantillon x_m de façon aléatoire dans une classe de données,
   - déterminer les k plus proches voisins ($x_1, x_2, \ldots, x_k$ pour $k \approx 10$) de x_m,
   - reconstruire un nouvel échantillon x'_m à partir d'une combinaison linéaire de ces k plus proches échantillons et de x_m (typiquement le centroide de ces $k+1$ points).
   - remplacer x_m et ces k voisins, par le nouvel échantillon x'_m.
3. **un filtrage statistique des *outliers***. Les vecteurs d'écart type $\boldsymbol{\sigma}_i$ et de moyenne $\boldsymbol{\mu}_i$ de la $i^{ème}$ classe, représentant respectivement la dispersion et le centroïde de cette classe dans l'espace des attributs, sont calculés. Un échantillon x_m de cette classe sera gardé à la condition que $\|x_m - \mu_i\|/\sigma_i < S_i$ est maintenue, où S_i est un seuil donné. Typiquement, si les attributs ont une distribution gaussienne, pour $S_i = 3$, nous savons qu'environ 2% des échantillons seront filtrés
4. **un sous-échantillonnage aléatoire de la base de données**. C'est la méthode la plus simple, qui, de manière surprenante, donne aussi des résultats acceptables et parfois supérieurs à l'approche précédente.

La méthode du *bootstrapping* a donné d'excellents résultats, en faisant une fusion ou agglomération des données autour d'un point x_i non préférentiel. Par contre, ni les techniques de filtrage statistique ni celles de classification par SVM ne fonctionnent mieux que la réduction aléatoire

des données. Nous pensons que ce phénomène s'explique par le fait que le filtrage statistique et le filtrage par SVM éliminent uniquement des échantillons bruités (ou aberrants) autour des frontières de séparation entre les classes, laissant une forte densité de données à l'intérieur du nuage de points de la classe.

3.5.2 Analyse Factorielle discriminante

Beaucoup des techniques de classification sont peu appropriées ou inefficaces à cause de la haute dimensionalité des données. L'analyse linéaire discriminante de Fisher (*LDA* fonction discriminante de Fisher) est une méthode bien connue et très utilisée dans la phase d'apprentissage d'un classifieur. Elle essaie de trouver un sous-espace dans lequel les moyennes des attributs des classes (centroïdes des nuages d'échantillons des classes) sont dispersées par rapport à la covariance interne de chaque classe (tailles des nuages). On doit projeter les données sur ce sous-espace, en évitant aussi des effets collatéraux comme des possibles pertes d'information ; le problème consiste alors à trouver la projection linéaire optimale ($y = w^t \cdot x$) qui satisfait tous ces critères.

Illustrons cette méthode pour deux classes définies dans un espace de deux attributs seulement. L'idée est de trouver une projection des échantillons, sur une droite qui sépare le mieux possible les deux classes. Le critère de séparation est exprimé par la maximisation du rapport des variances inter-classes et intra-classes dans cette projection. Autrement dit, la distribution de données d'une même classe doit être compacte, ce qui revient à minimiser la dispersion autour de la moyenne :

$$\widetilde{\boldsymbol{s}}_i^2 = \sum_{x_i \in C_i} (w^t \cdot x - \widetilde{\boldsymbol{\mu}}_i)^2 \,, \tag{3.26}$$

où la moyenne avant $\boldsymbol{\mu}_i$ et après projection $\widetilde{\boldsymbol{\mu}}_i$ sont données par :

$$\widetilde{\boldsymbol{\mu}}_i = \frac{1}{n_i} \sum_{x_i \in C_i} x \,, \qquad \widetilde{\boldsymbol{\mu}}_i = \frac{1}{n_i} \sum_{x_i \in C_i} \widetilde{\boldsymbol{x}} = \frac{1}{n_i} \sum_{x_i \in C_i} w^t \cdot x = w^t \cdot \mu_i \,. \tag{3.27}$$

Les noyaux (moyennes) des échantillons projetés des deux classes $\widetilde{\boldsymbol{\mu_i}}$ et $\widetilde{\boldsymbol{\mu}}_j$ devront se séparer le plus loin possible, ce qui est représenté par :

$$\widetilde{\boldsymbol{\mu}}_i - \widetilde{\boldsymbol{\mu}}_j = w^t \cdot (\mu_i - \mu_j) \,. \tag{3.28}$$

Le critère d'optimisation de Fisher pour séparer les deux classes est donné par :

$$J(w) = \frac{w^t \cdot \boldsymbol{S}_B \cdot w}{w^t \cdot \boldsymbol{S}_W \cdot w} \tag{3.29}$$

- le numérateur représente la matrice de dispersion intra-classe qui est obtenue à partir de $w^t \cdot \boldsymbol{S}_W \cdot w = \widetilde{\boldsymbol{s}}_i^2 + \widetilde{\boldsymbol{s}}_j^2$ où $\boldsymbol{S}_W$ est la somme des variances de chaque classe : $\boldsymbol{S}_W = S_i + S_j$, utilisant $S_i = \sum_{x_i \in C_i} (x - \mu_i)^t (x - \mu_i)$.
- le dénominateur représente la matrice de dispersion inter-classe qui est estimée par $w^t \cdot \boldsymbol{S}_B \cdot w = (\widetilde{\boldsymbol{m}}_i - \widetilde{\boldsymbol{m}}_j)^2$ où $\boldsymbol{S}_B = (m_i - m_j)^t \cdot (m_i - m_j)$.

La solution optimale de l'équation 3.29 étant donnée par :

$$w \propto \boldsymbol{S}_W^{-1} \cdot (\mu_i - \mu_j) \,, \tag{3.30}$$

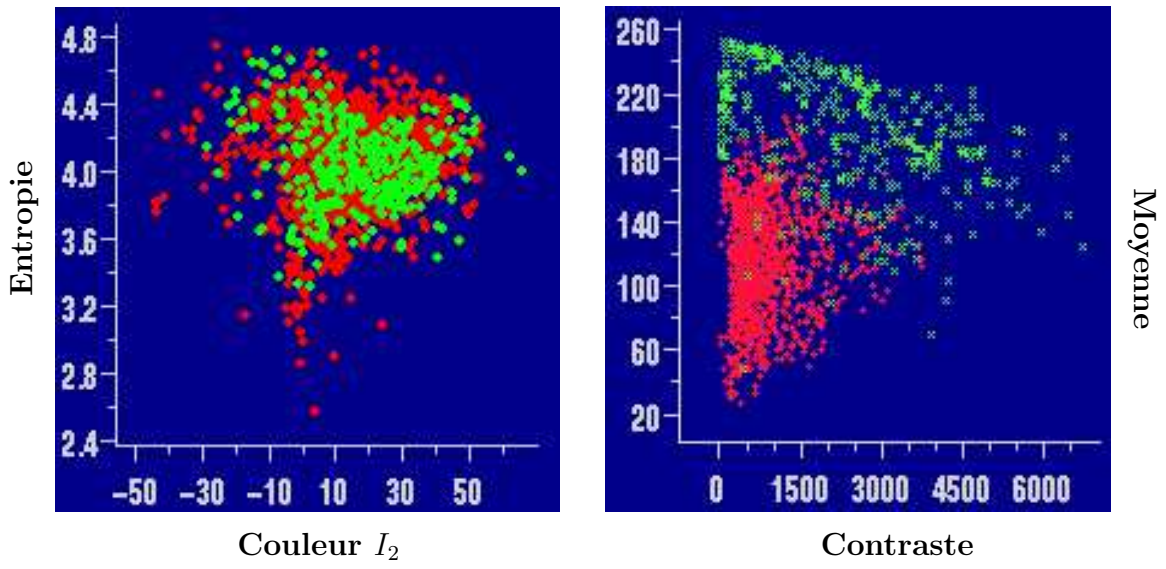


FIG. 3.10 – Nuage de points de la classe *arbre* et *haie*

où le vecteur propre w, est alors la droite qui sépare le mieux les deux classes.

L'utilisation de cette méthode est surtout fiable dans des applications où il n'y a pas un mélange important entre les classes, *i.e.*, elle est inefficace sur des problèmes non-linéairement séparables. Or, pour des applications en scènes naturelles cette approche n'est pas complètement valide car certains nuages de points sont partiellement mélangés. Par exemple, sur la figure 3.10, il est impossible de trouver une séparation « linéaire » entre les classes « arbre » et « haie », « haie » et « herbe haute », et entre « chemin » et « arbre ». Cela nous indique qu'une analyse discriminante non-linéaire serait plus appropriée.

3.5.3 Analyse en composantes principales (ACP)

L'analyse en composantes principales ACP[20] reste très utilisée dans les domaines du traitement des signaux, la statistique et les approches neuronales. L'analyse en composantes principales est souvent effectuée

- soit avant une régression afin d'éviter l'utilisation des attributs redondants,
- soit, éventuellement, avant une classification pour analyser la structure des échantillons afin de déterminer le nombre de classes à considérer.

Compte tenu de l' abondance des informations à traiter, le principal but de l'analyse en composantes principales est de représenter les données d'origine X de $\mathbb{R}^p$ en un nouveau nuage de points de dimension $q < p$, tel que les nouveaux attributs n'aient pas de corrélation entre eux et soient ordonnés en terme de la variance apportée par chaque composante $s_1, s_2, \ldots, s_q$. L'ACP consiste donc à obtenir ce sous-espace de dimension q, obtenu depuis $\mathbb{R}^p$, par une combinaison linéaire noté $\boldsymbol{W}_p$ des attributs d'origine.

Les composantes principales sont données par une projection $s_i = \boldsymbol{w}_i^t \boldsymbol{X}$. Si nous notons $\boldsymbol{w}_1$, la direction de la première composante cela revient donc à l'estimer de façon à ce qu'il vérifie :

$$\boldsymbol{w}_1 = \underset{\|\boldsymbol{w}\|=1}{\arg\min}\, E\{(\boldsymbol{w}^t \boldsymbol{x})^2\}\,, \tag{3.31}$$

[20] ACP, appelée aussi la transformation Karhunen-Löeve ou la transformation Hotelling, fut introduite par Karl Pearson en 1901 et intégrée à la statistique mathématique par Harold Hotelling en 1933.

en pratique le calcul des coefficients $\boldsymbol{w}_i$ exploite la matrice de covariance $\mathcal{C}$ calculée à partir des données d'origine. En effet, il a été démontré que la représentation donnée par l'ACP est une réduction de dimension « linéaire optimale » au sens de moindres carrés.

A partir de N échantillons décrits initialement par p attributs, les différentes opérations nécessaires pour l'ACP sont :

- une normalisation : les données (rangées en lignes) devront être centrées et réduites ($\boldsymbol{Y} = (\boldsymbol{X} - E\{\boldsymbol{X}\})/\boldsymbol{\sigma_X}$). Ceci est nécessaire du fait de la présence de différents types de grandeurs dans les attributs. La matrice Y a N lignes et p colonnes.
- le calcul de la matrice de corrélation ou de covariance des données normalisées : c'est une matrice carrée de dimension pxp, $\mathcal{C} = E\{\boldsymbol{Y}^t \cdot \boldsymbol{Y}\}$. Notons que l'analyse sur les matrices de covariance sur des données centrées et réduites, est identique à l'analyse sur des matrices de corrélation.
- le calcul des valeurs et vecteurs propres de la matrice de covariance, selon l'algorithme de décomposition en valeurs singulières SVD[21] [Stewart 93]. La SVD sur la matrice $\boldsymbol{Y}$ et celle sur la matrice de covariance $\mathcal{C}$ sont intimement liées ; en utilisant la SVD de $\boldsymbol{Y}$ dans l'expression de la covariance $\mathcal{C} = \boldsymbol{Y}^t \cdot \boldsymbol{Y} = (U \cdot D \cdot V^t)^t \cdot (U \cdot D \cdot V^t)$ et les propriétés des matrices orthonormées $U^t \cdot U = V^t \cdot V = \boldsymbol{I}$, la covariance est alors représentée par :

$$\mathcal{C} = V \cdot D^2 \cdot V^t \,, \tag{3.32}$$

  où V et D^2 sont respectivement les vecteurs et valeurs propres (λ_i) de $\boldsymbol{Y}^t \cdot \boldsymbol{Y}$ obtenus par la décomposition en valeurs singulières de $\boldsymbol{Y}$. Étant donné que le calcul de la matrice de covariance est coûteux et délicat, nous pouvons obtenir directement l'ACP (*cf.* éq. 3.32) en appliquant uniquement la SVD à $\boldsymbol{Y}$, ce qui est d'ailleurs très stable.
- les composantes principales $\boldsymbol{W}_p$ sont obtenues en ordonnant par colonnes les vecteurs propres donnés dans V selon l'ordre de la variance décroissante donnée par les valeurs propres λ_i données dans D^2. Le nombre de composantes est au plus égal à celui des attributs, mais l'utilisation pratique consiste à choisir q vecteurs propres tels que le rapport d'énergie donné par :

$$\sum_{i=1}^{q} \lambda_i / \sum_{i=0}^{p} \lambda_i \,, \tag{3.33}$$

  soit suffisamment proche de l'unité (*e.g.*, 98%).

Nous avons appliqué la méthode ACP à deux reprises, d'abord de manière classique, pour identifier les sept attributs les plus discriminants lors de la construction de notre base de données, et ensuite, dans le processus de classification par k-PPV en utilisant une métrique quadratique.

3.5.3.1 Utilisation de ACP pour réduire le nombre des attributs

Dans le processus de sélection des attributs nous avons analysé l'apport énergétique de chaque nouvel attribut ; par exemple, les attributs corrélés comme la moyenne et l'intensité seront à l'origine d'une valeur propre de très faible valeur témoignant de la forte corrélation entre ces deux attributs. L'un des deux devra être éliminé. En conséquence, nous avons retenu les attributs donnant une matrice diagonale D^2 dont les valeurs propres ont une énergie plus équilibrée (plus distribuée) sur l'ensemble des attributs. Les attributs de texture retenus qui

[21] D'après, le théorème de Young et Eckart, la décomposition en valeurs singulières pour une matrice X de dimensions $m \times n$ de rang $r \leq \min(m, n)$ est donnée par $X = U \cdot D \cdot V^t$ où U et V sont des matrices orthonormées formées respectivement des vecteurs propres de $\{X \cdot X^t\}$ et de $\{X^t \cdot X\}$, et D est une matrice diagonale formée avec les valeurs propres de X.

ont été utilisés dans la modélisation de l'environnement et la navigation visuelle, sont donc : l'entropie, le contraste, l'énergie, l'homogénéité, la corrélation, la nuance et la proéminence.

Par contre, nous aurions pu choisir comme attributs (*cf.* éq. 3.33) les q vecteurs propres $\boldsymbol{W}_p$ les plus discriminants, projeter nos données d'origine sur cette base réduite et classifier sur cet espace. Mais nous n'avons pas voulu

- ni caractériser des régions avec des attributs qui ont pas d'interprétation sémantique,
- ni ralentir le processus d'extraction des attributs et la reconnaissance avec une projection vers un autre espace de données.

Pour justifier ce choix, nous avons vérifié que la distance euclidienne entre deux échantillons dans l'espace d'origine (avant projection) ou dans l'espace réduit (après projection par ACP) donne quasiment le même résultat. Ce phénomène s'explique par le fait que l'ACP fait seulement une rotation des axes sans une modification de la structure des données, par exemple la distance entre deux vecteurs projetés $\langle y_i, y_j\rangle = \langle Mx_i, Mx_j\rangle = (Mx_i)^t \cdot (Mx_j) = x_i^t \cdot x_j = \langle x_i, x_j\rangle$ sur une base orthonormée M ne présente pas d'avantages pour la méthode k-PPV.

3.5.3.2 Utilisation de ACP dans le classifieur k-PPV

Nous avons dérivé néanmoins, une métrique pour la classification par k-PPV à partir des résultats de l'ACP. Nous utilisons une nouvelle base composée par les vecteurs propres V et les valeurs propres de X données par D, appelée la matrice des poids factoriels (« factor loadings ») $\boldsymbol{W}_L = DV^t$. Calculant à nouveau la distance euclidienne sur des vecteurs projetés sur cette base, nous avons

$$\langle y_i, y_j\rangle = \langle DV^t x_i, DV^t x_j\rangle = x_i^t \cdot VD^t DV^t X_j = x^t \mathcal{C} x$$

, où $\mathcal{C}$ est la matrice de corrélation. Cela est équivalent au fait de définir un produit scalaire

$$\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{C}} = \boldsymbol{x}^t \cdot \boldsymbol{\mathcal{C}} \cdot \boldsymbol{y}$$

, où $\boldsymbol{\mathcal{C}}$ est une matrice de pondération symétrique définie positive, avec une norme associée $\|x\|_{\mathcal{C}} = \sqrt{\langle x, x\rangle_{\mathcal{C}}}$. La métrique dérivée qui prend en compte la corrélation entre les attributs est alors donnée par l'équation suivante :

$$\boldsymbol{d}_{\mathcal{C}} = \sqrt{\boldsymbol{x}^t \cdot \boldsymbol{\mathcal{C}} \cdot \boldsymbol{y}}\,, \tag{3.34}$$

représentant une forme de distance quadratique dont la matrice de pondération est obtenue explicitement à partir de la base de données, et peut donc être pré-calculée dans la phase d'apprentissage [Hafner 95].

3.5.4 Analyse en composantes indépendantes (ACI)

L'analyse en composantes principales a révélé certaines limites dans le fait de simplement rechercher des variables décorrélées, elle ne suffit pas à assurer l'indépendance des variables dans des données de nature non gaussienne. De plus, l'ACP n'impose que des statistiques d'ordre deux et elle est déterminée par des bases orthogonales. Ainsi, nous nous sommes intéressés à l'utilisation de méthodes plus robustes qui pourraient résulter de l'application d'un principe de réduction de l'information redondante (améliorant l'étape de classification) par le principe d'indépendance statistique.

Issue du domaine de la séparation aveugle de sources[22], l'analyse en composantes indépendantes *ACI*[23] d'un vecteur aléatoire $\boldsymbol{X}$ de $\mathbb{R}^p$ consiste à rechercher une transformation linéaire qui réduit au minimum la dépendance statistique entre les composantes à la sortie. Utilisant des statistiques d'ordre supérieur, l'ACI est considérée comme la généralisation de l'ACP [Antonini 03]. En plus, la base de vecteurs d'une ACI capture mieux les caractéristiques locales dans l'espace d'attributs qu'une ACP, ce qui implique une meilleure représentation et reconnaissance des signaux.

L'ACI a attiré beaucoup d'attention en raison de ses applications potentielles dans le traitement des signaux médicaux (EEG et MEG), les systèmes de reconnaissance de la parole, les télécommunications, l'analyse financière, le traitement d'images et l'extraction d'attributs. Elle suppose que des signaux sont produits par plusieurs sources (indépendantes), dont les propriétés sont inconnues, supposant seulement un caractère non gaussien [Belouchrani 97].

3.5.4.1 Principe

Pour le vecteur $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)^t$, le problème revient à trouver une transformation $W \in \mathbb{R}^{n \times p}$ représentant le mélange d'un certain nombre n de *sources indépendantes*, données par un vecteur aléatoire $\boldsymbol{S} = (\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots \boldsymbol{S}_n)^t$, telles que :

$$\boldsymbol{X} = \mathcal{F}(\boldsymbol{S}) \tag{3.35}$$

où $\mathcal{F}$ est appelée *fonction de mélange*, $\boldsymbol{S}$ et $\boldsymbol{X}$ sont respectivement les signaux d'origine dits indépendants et les signaux mélangés. Il s'agit de trouver la meilleure façon de représenter les données $\boldsymbol{X}$ comme transformées des variables $\boldsymbol{S}$ par une fonction $\mathcal{F}$. Cette fonction peut nous conduire directement, soit à l'ACP si nous cherchons des composantes de variance maximale et non corrélées, soit à l'ACI si nous recherchons les sources statistiquement indépendantes qui permettent d'estimer la meilleure représentation des données.

Formellement, nous pouvons obtenir les sources par une fonction inverse $\mathcal{H}$ (fonction de séparation) donnée par $\boldsymbol{Y} = \mathcal{H}(\mathcal{F}(\boldsymbol{S}))$.

3.5.4.2 Définition

Sans perte de généralité, nous pouvons considérer le cas où les fonctions de mélange et de séparation sont des applications linéaires (*i.e.*, mélange additif et sans décalage de phase), elles sont exprimées alors sous la forme de matrices comme suit :

$$\boldsymbol{Y} = \boldsymbol{W} \cdot \boldsymbol{X} = \boldsymbol{W} \cdot \boldsymbol{A} \cdot \boldsymbol{S}\,, \tag{3.36}$$

où A est connue comme la matrice de melange (les colonnes $\boldsymbol{a}_i$ représentent des attributs) et $\boldsymbol{W}$ la matrice de séparation (*cf.* figure 3.11). La matrice $\boldsymbol{S}$ est composée des signaux $\boldsymbol{s}_i$ donnant l'amplitude du $i^{ème}$ attribut des données observées $\boldsymbol{X}$. Sous la condition que les composantes indépendantes $\boldsymbol{s}_i$ sont munies de variance unitaire, elles seront uniques aux signes près.

Il est possible aussi de définir d'autres transformations comme par exemple celles utilisant la convolution $\boldsymbol{X} = \boldsymbol{A} * \boldsymbol{S}$, une fonction post-non-linéaire $\boldsymbol{X} = \mathcal{F}(\boldsymbol{A} \cdot \boldsymbol{S})$ ou encore une fonction $\mathcal{F}$ non linéaire, mais nous ne nous intéresserons qu'à la transformation linéaire définie précédemment.

[22] La *Blind Source Separation* consiste à retrouver un certain nombre de sources à partir des observations d'un mélange de celles-ci, ne connaissant pas la manière dont les sources se mélangent, ainsi que le nombre de sources à retrouver (problème du *cocktail of signals*).

[23] L'ACI a été formulée explicitement par Hérault et Jutten et formalisée théoriquement par [Comon 94]

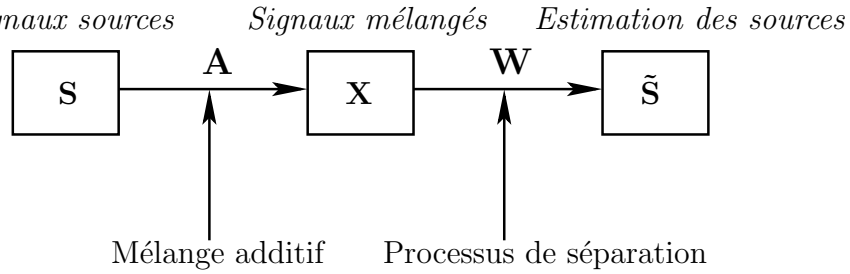


FIG. 3.11 – Processus impliqués dans la séparation des sources indépendantes

Il existe plusieurs approches pour estimer la matrice de séparation $\boldsymbol{W} \approx A^{\dagger}$ († pour désigner la matrice pseudoinverse) parmi lesquelles les plus représentatives sont :

- ⋆ une approche mesure l'indépendance entre les signaux en minimisant l'information mutuelle ou minimisant la neg-entropie par la métrique de l'information de *Kullback-Leibner* :

$$I(x) = J(x) - \sum J_i(x) = \int f(x) log \frac{f(X)}{\prod_i f_i(x)}\, dx\,, \tag{3.37}$$

  où $J(x)$ représente la neg-entropie qui prend en compte la non-gaussianité du signal :

$$J(x) = H(x_{gauss}) - H(x) \text{ où l'entropie est définie par } H(x) = -\int f(x) log(f(x))\, dx \tag{3.38}$$

  Dans cette représentation les $f_i(x)$ sont la densité de probabilité d'un vecteur aléatoire $\boldsymbol{S}$ et x_{gauss} est une variable aléatoire gaussienne ayant la même matrice de covariance que x.
- ⋆ une deuxième approche consiste à annuler tous les moments $\mu_r(x) = E\{(x - E\{x\})^r\}$ et les cumulants

$$Cum(x_i, \ldots, x_n) = \frac{1}{k!} \frac{\partial}{\partial u_i} \cdots \frac{\partial}{\partial u_n} ln \left(E\{e^{\sum_i u_i x_i}\} \mid_{u_1 \ldots u_n = 0} \right) \tag{3.39}$$

  pour trouver l'indépendance statistique. Notons que les trois premiers cumulants sont égaux aux moments correspondants. Le cumulant de quatrième ordre est également connu comme *kurtosis*. Pour un processus gaussien tous les cumulants d'ordre supérieur à deux sont zéro. Le kurtosis est généralement employé alors comme mesure de non gaussianité d'une variable aléatoire.
- ⋆ le troisième type d'approches exploite directement la définition de Kurtosis

$$\kappa_4 = E\{x^4\} - 3(E\{x^2\})^2 \tag{3.40}$$

  pour l'estimation des composantes indépendantes. Cet algorithme a été proposé originalement par Hyvärinen comme une alternative pour la minimisation de l'information mutuelle [Yuen 02].

Plus précisément, l'algorithme que nous avons adopté est appelé *Fast-ICA* [Hyvärinen 99], profitant de ses propriétés de convergence rapide et d'extraction séquentielle (déflation) ou parallèle de composantes indépendantes. C'est une méthode qui bien que dépendant de la dimension de la base de données, peut être utilisée dans des applications en temps réel. En bref, le problème revient à trouver une « mesure de non-gaussianité » utilisant des fonctions de contraste pour estimer les sources indépendantes de façon rapide par la méthode itérative de point fixe.

3.5.4.3 Méthode itérative de point fixe

La procédure itérative pour obtenir les composantes indépendantes d'un ensemble de données réelles, comprend fondamentalement les opérations suivantes :

1. de la même manière que pour l'ACP, un centrage des données : $\boldsymbol{Y} = \boldsymbol{X} - E\{\boldsymbol{X}\}$,
2. le blanchissement de données (*whitening* ou *esfering*), nous utilisons une transformation linéaire $\boldsymbol{Z} = \boldsymbol{Q} \cdot \boldsymbol{Y}$ pour décorréler et annuler les statistiques d'ordre deux ($E\{\widetilde{\boldsymbol{Z}}^t \cdot \widetilde{\boldsymbol{Z}}\} = \boldsymbol{I}$) dans des vecteurs dits blanchis $\widetilde{\boldsymbol{Z}}$. Les matrices de blanchissement $\boldsymbol{Q}$ et de déblanchissement $\boldsymbol{Q}^{-1}$ sont obtenues à l'aide d'une ACP (*cf.* équation 3.32). Cette opération est représentée par les relations suivantes :

$$Q = D^{-1} \cdot V^t \qquad\qquad Q^{-1} = V \cdot D^{-1}\,. \tag{3.41}$$

3. fournir une matrice d'initialisation W de composantes indépendantes. Elle est généralement obtenue en générant aléatoirement p vecteurs orthonormés,
4. l'orthonormalisation symétrique de composantes, $W' = (WCW^t)^{1/2}W$ et l'estimation par une méthode itérative introduisant des non-linéarités g (kurtosis),

$$w' = w - \mu \frac{E\{z \cdot g(w^t z)\} - \beta w}{E\{g'(w^t z) - \beta}\,, \tag{3.42}$$

où g est la dérivée d'une *fonction de contraste* G (introduction d'une non linéarité) utilisée par le critère d'indépendance statistique (neg-entropie). Dans chaque itération les vecteurs sont orthonormalisés à nouveau, jusqu'à la convergence. Ce processus itératif est une variante de la méthode Newton-Raphson dont le paramètre μ joue le rôle de modificateur de la vitesse de convergence.

Dans [Hyvärinen 99], le lecteur intéressé peut consulter en détail les possibles alternatives pour le choix de la fonction de contraste G.

Hyvärinen montre que la vitesse de convergence de cette méthode itérative du point fixe est clairement supérieure à celles des approches neuronales. Ainsi, *fast-ICA* est considéré comme un algorithme général rapide qui permet l'optimisation séquentielle ou parallèle des fonctions de contraste.

Bien qu'il y ait beaucoup d'applications potentielles de l'ACI, nous allons spécifiquement nous intéresser à l'utilisation de l'ACI dans le pré-traitement et la classification des données issues des images du monde réel, *i.e.*, les attributs caractérisant les régions préalablement segmentées et caractérisées dans la scène.

3.5.4.4 Utilisation de ACI pour des tâches de classification

La détection et la reconnaissance de visages par ACI a été déjà explorée par Pong [Yuen 02], reportant des améliorations dans le taux de reconnaissance par rapport à l'ACP. Antonini a conçu une intéressante architecture de classification exploitant la puissance de la technique des *Support Vector Machines (SVM)* [Vapnik 96] [24] et le pré-traitement des données par ACI [Antonini 03] pour l'identification et la localisation robuste de visages dans des mauvaises conditions d'illumination ; les résultats obtenus sont encourageants.

Par contre, d'autres auteurs [Draper 03] n'ont pas trouvé de différences flagrantes dans les résultats de classification en appliquant, soit une ACI, soit une ACP lors de la phase d'apprentissage. Appliquée à des images naturelles, l'ACI permet de faire émerger les structures

[24] Nous avons renoncé à traduire SVM en français : 'machine à vecteurs support' ne nous semble pas pertinent.

fondamentales de celles-ci (*e.g.*, les bords, l'atténuation du bruit) qui peuvent être exploitées dans des processus de reconnaissance d'objets.

Une manière d'appliquer l'ACI consiste à considérer qu'une image est représentée par la superposition linéaire de N fonctions de base $\Phi_i(x, y)$ (estimées par la matrice $\boldsymbol{A}$) pondérant les sources $(s_1, \ldots, s_N)$ indépendantes. Dans les travaux sur le visage, ce modèle n'est appliqué que sur une imagette (une portion de l'image) contenant l'objet d'intérêt $I_m(x, y) = \sum_{i=1}^{N} \Phi_i(x, y) s_i$. Les données mélangées X sont formées par les imagettes collectées, dépliées et accolées les unes aux autres, comme pour l'ACP. Un algorithme d'extraction de composantes indépendantes est ensuite appliqué sur ces données estimant la matrice de séparation $\boldsymbol{W}$ pour estimer les sources indépendantes $(\widetilde{\boldsymbol{s}}_1, \ldots, \widetilde{\boldsymbol{s}}_N)$ [Yuen 02]. La matrice $\boldsymbol{W}$ contient les descripteurs recherchés sur chaque ligne.

Dans notre application, nous ne travaillons pas sur des imagettes, mais sur un ensemble de vecteurs de $\mathbb{R}^p$ préalablement catalogués par classe c_i lors d'un processus d'apprentissage supervisé. Tout d'abord, les vecteurs sont rangés dans un tableau de données de dimension $n \times p$. Les données subissent plusieurs pré-traitements (données centrées, blanchies et orthonormalisées) avant d'être utilisés comme paramètres d'entrée dans l'algorithme *Fast-ICA*, afin de les rendre mieux conditionnées (élimination de singularités). Dans les paramètres utilisés dans l'algorithme de Hyvärinen, nous utilisons une fonction de contraste[25] cubique $g(y) = y^3$ ce qui est équivalent à utiliser la définition de *Kurtosis* comme critère lors de l'obtention des composantes indépendantes. La sélection de la fonction de contraste est basée principalement sur un critère de complexité algorithmique, *i.e.*, nous avons retenu celle qui prend le moins de temps de calcul.

Nous avons remarqué qu'une obtention de composantes indépendantes en parallèle (ou symétrique) est généralement beaucoup plus rapide qu'une approche par déflation (calcul séquentiel des composantes). De plus, nous avons choisi cette approche parallèle car celle par déflation semblait accumuler des erreurs d'estimation pendant la convergence. Dans ces conditions, pour nos bases de données, nous n'avons pas trouvé de problèmes particuliers liés à la convergence. En conséquence, nos vecteurs sources sont obtenus par leur projection sur la base W par la relation suivante :

$$\boldsymbol{s} = \boldsymbol{W} \left(x - E\{x\}\right) . \tag{3.43}$$

Les vecteurs inconnus projetés peuvent être classifiés en les comparant avec les vecteurs sources obtenus par chaque classe. Dans nos expériences de classification, nous avons entraîné une méthode de SVM avec les vecteurs dits indépendants obtenus depuis notre base de données par une ACI ; plusieurs de ces résultats sont discutés en § 3.6.1 ainsi que l'amélioration de la performance impliquée par l'utilisation d'une ACI pour améliorer la base d'apprentissage.

3.6 Identification et classification des régions

Après avoir subi les pré-traitements évoqués au chapitre 2, les images couleur sont segmentées pour extraire les principales régions dans la scène. Pour caractériser des régions, nous utilisons le vecteur d'attributs dans $\mathbb{R}^{12}$, calculé à partir de la texture et de la couleur et enrichi avec des informations contextuelles de position.

Les méthodes de reconnaissance utilisent l'information contenue dans une base d'apprentissage, dans laquelle se trouvent la liste des classes des entités à identifier dans les images, et les échantillons représentatifs de chaque classe. Cette base est construite en mode supervisé et

[25] D'autres fonctions de contraste sont aussi disponibles : $G_1(u) = log\, cosh(a_1 u)$, $G_2(u) = exp(-a_2 u^2/2)$ et $G_3(u) = u^4$ où $a_1, a_2 \geq 1$ sont des constantes à régler.

mise à jour en continu (cf § 3.5.1). Dans la suite, nous décrirons les méthodes de classification de régions qui ont été implémentées (à savoir SVM et k-PPV) ainsi que deux autres techniques souvent citées dans la littérature, mais que nous n'avons pas essayées. Nous évoquons ensuite comment des informations contextuelles peuvent être prises en compte.

3.6.1 Support Vector Machines

Depuis quelques années, les *Support Vector Machines* (SVM) ont émergé, comme une nouvelle technique d'apprentissage supervisé pour la classification de données. Boser, Guyon et Vapnik [Boser 92] ont proposé le premier classifieur binaire de cette catégorie qui a par ailleurs été étendu à des problèmes de régression. La méthode SVM est basée sur le principe de la minimisation de la fonction de risque structurel [Vapnik 98].

Cette méthode a pour objectif de rechercher le meilleur hyperplan $(\boldsymbol{w}^T \cdot \boldsymbol{x} + b)$ de séparation des données en deux classes. La classification d'un nouvel individu x de $\mathbb{R}^n$ est donnée par sa position par rapport à cet hyperplan, *i.e.*, le signe donné en substituant X dans l'équation de l'hyperplan. Les solutions obtenues par SVM sont indépendantes de la dimension des données et ne tombent jamais dans des minima locaux. Que ce soit en classification ou en régression, les SVM ont montré d'excellentes performances pour une grande variété d'applications [Reyna-Rojas 02, Qi 01, Li 03]. Nous présenterons un rappel théorique des SVM et leur mode d'emploi pour traiter des problèmes de classification.

3.6.1.1 Principes théoriques

Considérons un problème de classification à partir d'un ensemble de données d'apprentissage $D = \{\boldsymbol{x_i}, y_i\}_{i=1}^{k}$ où chaque entrée est un couple constitué par un échantillon $\boldsymbol{x_i} \in X \in \mathbb{R}^n$ et l'étiquette de sortie $y_i \in \{\pm 1\}$ (2 classes A et B). Dans un premier temps, la méthode SVM transforme un vecteur $\boldsymbol{x}$ en $\boldsymbol{z} = \phi(\boldsymbol{x}) \in \mathcal{F} \in \mathbb{R}^M$ dans un espace de Hilbert $\mathcal{F}$ (muni d'un produit scalaire$\langle \cdot, \cdot \rangle$) utilisant une fonction non-linéaire $\phi : \mathbb{R}^n \rightarrow \mathcal{F}$. Cet espace $\mathcal{F}$ est appelé l'*espace de caractéristiques ou d'attributs*, qui est généralement de dimension M très élevée (et parfois infinie). L'hypothèse utilisée pour classifier des échantillons inconnus, consiste à construire dans $\mathcal{F}$ des hyperplans de séparation à partir des données d'apprentissage (voir figure 3.12).

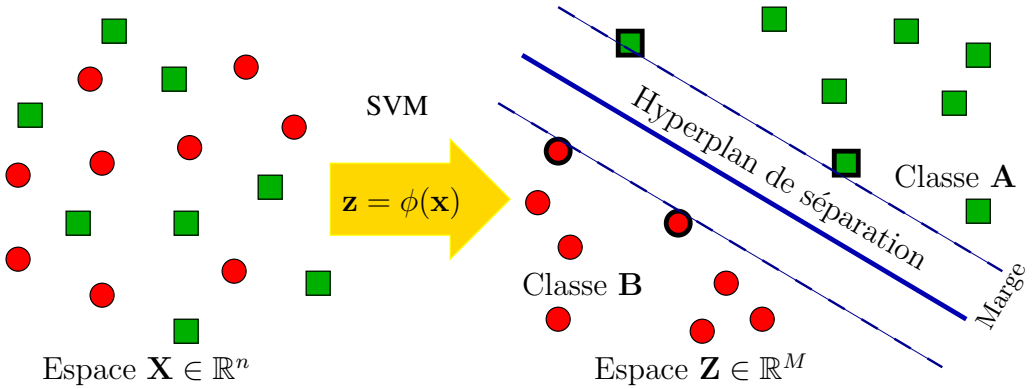


FIG. 3.12 – Séparation de classes par SVM, en transformant un problème dans un espace $\boldsymbol{X}$ de $\mathbb{R}^n$ en un problème dans un autre espace $\mathcal{F} \in \mathbb{R}^M$ de dimension beaucoup plus élevée. Les vecteurs support sont entourés en trait gras.

Une machine linéaire est construite en utilisant le concept de perceptron classique, représenté

par la fonction linéaire :

$$f(\boldsymbol{z}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle + b = \sum_{i=1}^{M} w_i \phi_i(x) + b \tag{3.44}$$

où $\boldsymbol{w} \in \mathbb{R}^M$ est un vecteur de « poids » et, en définissant les hyperplans avec les paramètres $b \in \mathbb{R}$. L'hyperplan de séparation optimal est estimé en maximisant les distances perpendiculaires (signées) entre les points d'entraînement et cet hyperplan, *i.e.*, en minimisant $\|\boldsymbol{w}\|$. La solution est donnée par $\boldsymbol{w} = \sum_{i=1}^{k} \alpha_i y_i \phi(\boldsymbol{x_i})$ pour des paramètres $\alpha_i \geq 0$. Le vecteur $\Lambda = (\alpha_1, \ldots, \alpha_k)$ peut être calculé par une méthode d'optimisation quadratique (QP), ce qui s'exprime par :

$$\text{maximiser} \quad W(\Lambda) = \Lambda^t \mathbf{1} - \frac{1}{2} \Lambda^t \boldsymbol{Q} \Lambda \,, \tag{3.45}$$

soumise sous les contraintes, $\Lambda \geq 0$ et $\Lambda^t \boldsymbol{Y} = 0$. Ici, Q est une matrice symétrique composée par des éléments $Q_{ij} = y_i y_j \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) \rangle$ et $\boldsymbol{Y}^t = (y_1, \ldots, y_k)$.

Les échantillons d'apprentissage correspondants aux $\alpha_i \geq 0$ qui vont être attachés aux marges[26] de la frontière de décision par le théorème de Karush-Kuhn-Tucker (généralisation des multiplicateurs de Lagrange) seront appelés les *vecteurs support* [Vapnik 96].

Étant donné que la dimension de $\mathcal{F}$, de $\phi(\boldsymbol{x_i})$, de $\phi(\boldsymbol{x_j})$... est très élevée et prohibitive dans des algorithmes réels, la méthode SVM a dû utiliser une stratégie pour éviter de calculer explicitement les transformations $\phi(\boldsymbol{x})$ et pour obtenir en même temps les produits scalaires $\langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) \rangle$. Cette stratégie constitue le principal objectif du développement des SVM ; elle s'appuie sur le fait que pour n'importe quel algorithme trouvant des solutions non linéaires en $\mathcal{F}$, si l'algorithme n'exécute que des opérations scalaires en $\mathcal{F}$, l'utilisation des méthodes basées sur des fonctions noyau (ou *kernels*) éviteront alors tout calcul explicite de ces transformations $\phi(\boldsymbol{x})$.

Ainsi, pour transformer les points de l'espace d'entrée $X \; in \mathbb{R}^n$ dans l'espace des caractéristiques $\mathcal{F}$, on recherche le produit interne dans l'espace de Hilbert via des fonctions noyau du type :

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \langle \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) \rangle \,. \tag{3.46}$$

De plus, en utilisant les fonctions noyau, on peut se permettre de prendre en compte les statistiques de plus grand ordre, vu l'absence d'une explosion combinatoire de la complexité que l'on aurait rencontrée sans cette notion. En général, il est possible d'utiliser n'importe quelle fonction noyau [Autio 03] en satisfaisant uniquement le théorème de Mercer [Vapnik 98] ; les fonctions noyau les plus utilisées sont les suivantes :

1. le noyau linéaire,
$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i}^t \cdot \boldsymbol{x_j} \tag{3.47}$$
2. les polynômes de degré d,
$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i}^t \cdot \boldsymbol{x_j} + 1)^d \tag{3.48}$$
3. les fonctions à base radiale (RBF),
$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2}{2\sigma^2} \right) , \tag{3.49}$$
où σ contrôle la largeur du noyau gaussien

[26] la marge est la distance d'un point x à l'hyperplan de séparation, *i.e.*, marge $= | \boldsymbol{w} \cdot \boldsymbol{x} + b | / \|w\|$. L'hyperplan optimal est celui ayant la norme de coefficients la plus petite (la plus grande marge).

4. les réseaux de neurones (MLP) à deux couches,

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = tanh\left(\kappa(\boldsymbol{x_i}^t \cdot \boldsymbol{x}_j + \delta)\right) , \tag{3.50}$$

où κ et δ doivent vérifier l'inégalité $\kappa > \delta$.

Dans ces expressions, σ, d, κ et δ sont appelés les paramètres du noyau. L'utilisation des fonctions noyau est une manière élégante de résoudre un problème d'optimisation quadratique et les solutions obtenues ne sont jamais des solutions locales.

3.6.1.2 Classification par SVM

Dans le domaine de la reconnaissance d'objets en les comparant aux échantillons de la base d'apprentissage, la méthode de classification exploite donc les vecteurs support de marge optimale, qui déterminent l'hyperplan de séparation dans l'espace $\mathcal{F}$. Cet hyperplan maximise l'équation suivante, obtenu à partir de l'équation vectorielle 3.45 :

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{k} y_i y_j \alpha_i \alpha_j \langle x_i, y_i \rangle , \tag{3.51}$$

soumise aux contraintes $\sum_{i=1}^{k} y_i \alpha_i = 0$ et $\alpha_i \geq 0$. Notons $\boldsymbol{\alpha}^*$ et b^*, les solutions de ce problème d'optimisation. Les vecteurs support sont les vecteurs d'entrée $\boldsymbol{x_i}$ les plus proches de l'hyperplan ayant des α_i^* non nuls. Si ces vecteurs sont regroupés dans un ensemble noté $\mathcal{V}$, la marge géométrique à l'hyperplan de séparation, est définie par : $\gamma = (\sum_{i \in \mathcal{V}} \alpha_i^*)^{1/2}$.

La règle de décision dérivée de ces vecteurs support, est donnée par le « signe » de la fonction d'activation suivante :

$$f(\boldsymbol{z}) = \sum_{i=1}^{k} y_i \alpha_i^* \langle x_i, \phi(x) \rangle + \boldsymbol{b}^* = \sum_{i=1}^{k} y_i \alpha_i^* K(\boldsymbol{x_i}, \boldsymbol{x}) + \boldsymbol{b}^* , \tag{3.52}$$

où $\boldsymbol{x} \in \mathbb{R}^n$ est le vecteur à classifier.

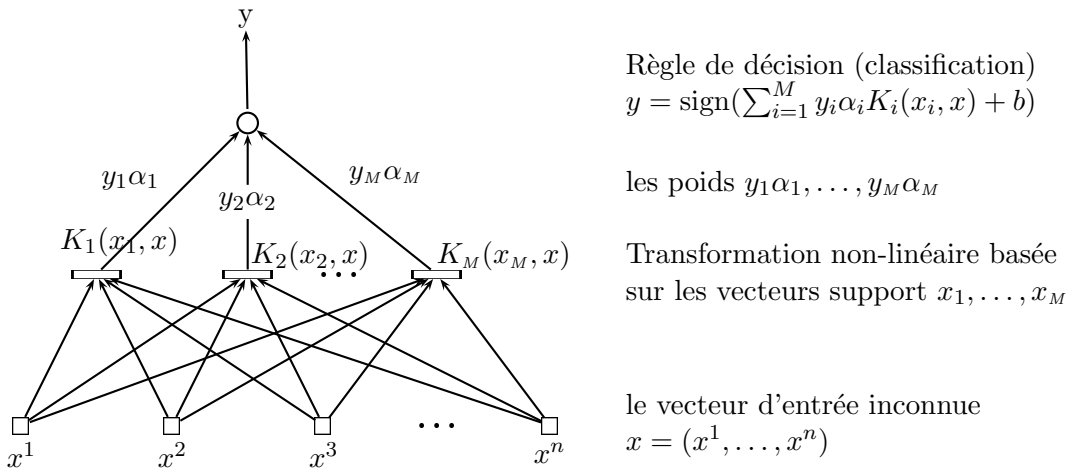


FIG. 3.13 – Diagramme de la machine à vecteurs support.

En bref, la figure 3.13 présente un diagramme montrant les couches primordiales d'une méthode SVM. Elle est essentiellement composée de deux couches. Pendant le processus d'apprentissage, la première couche fait la sélection de la meilleure base $K(\boldsymbol{x_i}, \boldsymbol{x}), i = 1, \ldots, M$ dans l'ensemble des bases définies par la fonction noyau ; la deuxième couche construit une fonction linéaire dans cet espace qui sert à la classification. Cela est équivalent à construire l'hyperplan optimal dans l'espace d'attributs correspondant.

3.6.1.3 Des échantillons non séparables

La classification en utilisant la notion de marge maximale requiert des données linéairement séparables. Par contre, les données recueillies dans des applications réelles sont généralement bruitées ce qui produit le mélange spatial des données entre deux classes différentes.

Quand les données d'apprentissage dans l'espace $\mathcal{F}$ sont non séparables, les SVM essaient de séparer les deux classes en utilisant des contraintes moins rigides dans le problème d'optimisation. Ainsi, la méthode SVM essaie de déterminer la minimisation de $\|\boldsymbol{w}\|$ permettant d'aboutir à une séparation de classes avec la quantité d'erreurs moindre. Cela est exprimé par l'équation suivante :

$$J = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=0}^{k}\xi_i\,, \tag{3.53}$$

où $\xi_i \geq 0$ doit satisfaire la contrainte : $y_i(\langle \boldsymbol{x}, \phi(\boldsymbol{x_i})\rangle) \geq 1-\xi_i$. C est un paramètre de régularisation contrôlant l'erreur d'apprentissage.

3.6.1.4 Caractéristiques des SVM

Les SVM présentent une bonne résistance au problème de sur-apprentissage, d'où leur robustesse pour traiter de tâches de classification. Ils s'avérent particulièrement efficaces par le fait qu'ils peuvent traiter des problèmes mettant en jeu de grands nombres de descripteurs. Les propriétés des fonctions noyau assurent une procédure d'optimisation convexe du problème des SVM ce qui permet l'obtention d'une solution optimale unique (pas de problèmes de minimum local).

Par contre, les SVM ont certains points faibles. La phase d'apprentissage reste particulièrement coûteuse en temps de calcul. En particulier, il n'est pas facile de développer une version multi-classes robuste : la méthode SVM permet de dire si un objet inconnu appartient ou non à une classe ; si un objet peut appartenir à plusieurs classes (par exemple, une zone de terrain qui peut être classée *Eaux*, *Chemin*, *Champ*...) il faut entraîner la machine pour chacune de ces classes. De plus, le critère de sélection de la meilleure fonction noyau pour une application donnée reste parfois arbitraire.

3.6.1.5 Application à la modélisation de l'environnement

Bien que les SVM puissent être appliqués directement sur les données d'origine, l'exploitation d'attributs ayant des grandeurs très différentes peut être responsable de la faible convergence de l'algorithme. Nous avons donc préféré utiliser une base de données normalisée. À cet effet, nous déterminons le vecteur composé par les valeurs maximales (en valeurs absolues) de chaque attribut dans toute la base d'apprentissage. Ensuite, tous les échantillons de la base de données sont divisés par ce vecteur d'attributs maximaux,

$$\boldsymbol{xnorm_i} = \left(\frac{\boldsymbol{x}_i^1}{\boldsymbol{x}_{max}^1} \ldots \frac{\boldsymbol{x}_i^k}{\boldsymbol{x}_{max}^k} \ldots \frac{\boldsymbol{x}_i^n}{\boldsymbol{x}_{max}^n}\right)\,, \tag{3.54}$$

où $\boldsymbol{x}^k$ représente ici le $k^{ème}$ attribut du vecteur $\boldsymbol{x}$.

Ces données normalisées vont alimenter la phase d'apprentissage de la méthode SVM. Après convergence, les vecteurs support obtenus nous permettront de disposer d'une fonction d'activation pour réaliser la classification des données inconnues. Il nous reste à choisir la stratégie à suivre lorsque nous disposons d'un classifieur SVM binaire dans un problème d'apprentissage multi-classes. En effet, il existe deux alternatives :

1. *une approche hiérarchique*. D'abord, nous effectuons l'apprentissage SVM en comparant l'une des classes C_1 par rapport aux $(k-1)$ classes restantes. Une base de données réduite est obtenue sans la classe C_1. Ensuite, nous relançons la procédure d'apprentissage pour comparer une autre classe C_2 par rapport aux autres $(k-2)$ classes de la base de données réduite (*i.e.*, sans les classes C_1 et C_2). Ce processus continue jusqu'à avoir uniquement deux classes. Ceci nous permet alors de construire un arbre de décision lors de l'étape reconnaissance.
2. *un approche parallèle* (« un contre le reste »). Pour chacune des classes C_i, nous lançons l'algorithme d'apprentissage en comparant la classe C_i avec les $k-1$ classes restantes.

La première approche fonctionne très bien dans des situations qui requièrent la recherche systématique de toutes les classes sur la scène, mais elle ne donne aucune mesure de confiance sur les résultats obtenus.

La seconde approche s'avère la plus rapide, permettant la détection individuelle des classes tout en évitant le balayage complet d'un arbre de décision. Par exemple, cela s'avère très utile dans la détection spécifique de la classe *Chemin*. Nous avons exploité particulièrement la deuxième approche même si cela pourrait en principe nous donner des résultats contradictoires (*i.e.*, étiquetage multiple) dans la reconnaissance globale de la scène. En effet, nous considérons que si un échantillon aberrant est classifié dans plusieurs classes, nous utiliserons la classification donnant *la marge maximale* (distance) de l'échantillon inconnu à l'hyperplan de séparation. Une mesure d'erreur (ou de confiance) de classification peut être obtenue en exploitant les amplitudes des fonctions d'activation d_i, (*i.e.*, $\eta = d_i / \sum d_i \mid d_i \geq 0$).

De plus, la classification multiple d'un objet peut être corrigée en utilisant une approche contextuelle qui prendrait en compte les relations structurelles des objets sur la scène.

3.6.1.6 Utilisation de l'ACI avec la méthode SVM

Étant donné que le processus d'apprentissage devient plus lourd quand les données sont bruitées ou mal conditionnées, nous avons estimé pertinent de combiner la puissance d'une méthode *Support Vector Machine* à une analyse en composantes indépendantes. En effet, le SVM sous sa forme initiale revient à chercher une frontière de décision linéaire entre deux classes, mais ce modèle peut considérablement être enrichi en se projetant dans un autre espace permettant d'augmenter la séparabilité des échantillons. Nous pouvons alors appliquer l'apprentissage dans ce nouvel espace, ce qui se traduit par une frontière de décision non linéaire dans l'espace initial.

Nous avons donc étudié une approche hybride SVM/ACI dans le cadre de la classification des régions dans des images naturelles d'extérieur. En § 3.7, nous discuterons plusieurs résultats et conclurons sur l'exploitation de cette approche SVM/ACI dans notre contexte.

3.6.2 La méthode des K-plus proches voisins

Parmi les méthodes de classification, la technique des k plus proches voisins (noté k-PPV) (ou *Case-Based Classification*) est particulièrement classique. Ainsi, l'approche k-PPV offre l'avantage d'être une technique très simple mais puissante dédiée principalement à la classification et

qui peut être étendue à des tâches d'estimation. Elle s'appuie sur le principe de prendre des décisions en mesurant la similarité d'un élément inconnu avec les échantillons déjà stockés et classés. Ainsi, on doit simplement chercher les étiquettes de classe d'un certain nombre k de *plus proches voisins* et en fonction de ces voisins, prendre une décision pour assigner une étiquette à cet élément inconnu.

Cette méthode diffère des traditionnelles méthodes d'apprentissage car aucun modèle n'est appris à partir des échantillons. Les données après normalisation, sont simplement stockées en mémoire. Le nombre de plus proches voisins, k, doit être impair afin d'éviter les ambiguïtés et il doit être maintenu petit, puisqu'un grand nombre k tend à créer de fausses classifications à moins que les classes individuelles ne soient bien séparées.

Il est facile de vérifier que le résultat global de ce type de classifieurs est toujours au moins la moitié de celui obtenu par le meilleur classifieur pour un problème donné. Par contre, un des inconvénients principaux de ces méthodes se situe dans l'étape d'apprentissage, *i.e.*, le classifieur a besoin de toutes les données disponibles. Ceci peut entraîner une augmentation de la complexité de calcul et donc, du temps de classification, si la base de données est considérablement grande. En effet, contrairement aux autres méthodes de classification (*SVM*, arbres de décision, réseaux de neurones, algorithmes génétiques...), il n'existe pas d'étape proprement dite d'apprentissage consistant à construire un modèle compact à partir des échantillons de la base de données.

L expérience montre par ailleurs que cette méthode présente souvent un bon pouvoir prédictif [Duda 98].

3.6.2.1 Description de l'algorithme

La première étape dans la méthode des k-PPV consiste à normaliser les données de $\mathbb{R}^n$, dans l'espace des attributs, afin de générer des vecteurs avec des attributs de grandeurs équivalentes. Cela revient a diviser les vecteurs $\boldsymbol{x}_i$ par le vecteur des attributs maximaux $\boldsymbol{x}_{max}$, de manière identique à ce qui est fait pour les SVM. Nous avons également testé une normalisation centrée et réduite mais les résultats de classification sont de qualité moindre.

Un paramètre important de k-PPV consiste à choisir le nombre d'échantillons k qui seront impliqués dans la tâche de décision. Dans notre application, ce paramètre dépend exclusivement de la dimension des données ; typiquement, le nombre k est donnée par la racine carrée[27] de la cardinalité de chaque classe ($k_i = \sqrt{n_c}$ avec $n_c = \#classe_i$).

Ensuite, la méthode nécessite une métrique pour mesurer la distance entre l'échantillon à classifier z et les vecteurs de la base d'apprentissage x. Ainsi, la distance de *Minouski* est la représentation la plus générale qui est adaptée aux problèmes de classification avec k-PPV. Elle est donnée par :

$$L_p = \{\sum_{i=1}^{n} \mid z_i - x_i \mid^p\}^{1/p} \tag{3.55}$$

où les normes L_1 et L_2 sont respectivement la distance « city-block » et la distance euclidienne.

Nous préférons utiliser une métrique qui tient compte des rapports intrinsèques entre les variables ou les attributs ; il s'agit d'un cas particulier des distances quadratiques [Hafner 95] de forme semblable à la distance de Mahalanobis. Cette métrique est présentée en § 3.5.3 et est donnée par l'équation 3.34. Dans cette équation la matrice de pondération est donnée par la matrices de coefficients de corrélation, calculée sur les données centrées et réduites. En outre, la métrique prend en compte la corrélation entre les différents attributs.

[27] bien d'autres choix sont possibles, *e.g.*, $k = ln(n_c)$

La classification revient alors à identifier à quelle classe c appartiennent les k_n échantillons les plus proches de l'échantillon inconnu $\boldsymbol{z}$ en utilisant notre critère de distance. $\boldsymbol{z}$ sera affecté à la classe la plus représentée parmi ces k_n vecteurs.

3.6.2.2 Estimation de probabilités de classification

On peut définir un estimateur de la densité de probabilité $f(x)$ tout en supposant que le volume V_k d'une région R_k centrée sur x augmente jusqu'à ce qu'il contienne les k_n échantillons plus proches. Cet estimateur est défini par :

$$\widehat{\boldsymbol{fn}}(x) = \frac{k_n/n}{V_n} \tag{3.56}$$

où n est le nombre de points et k_n est donné par $\sqrt{n}$. Les conditions nécessaires et suffisantes pour estimer $\widehat{\boldsymbol{f}}(x)$ sont :

$$f_n(x) \to f(x) \quad \text{si} \quad \lim_{k\to+\infty} k = +\infty \quad \text{et} \quad \lim_{k\to+\infty} k/n = 0 \tag{3.57}$$

Cela nous permet de calculer la probabilité d'avoir un vecteur x_i dans une région R centrée sur x, *i.e.*, $P_{\scriptscriptstyle R}(x) = \int_R f(u)du$.

Nous avons la possibilité d'estimer de façon directe $P(w|\boldsymbol{x})$ en utilisant $P_n(\boldsymbol{x}, \omega_i) = \frac{k_i/n}{V} = P(\omega_i)p_n(\boldsymbol{x}|\omega_i)$ et la règle de décision de Bayes,

$$p_b(\omega_i|\boldsymbol{x}) = \frac{P(\omega_i)p_n(\boldsymbol{x}|w_i)}{\sum_{j=1}^{c} P(\omega_i)p_n(\boldsymbol{x}|w_i)} = \left(\frac{k_i/n}{V}\right) / \sum_{j=1}^{c}\left(\frac{k_j/n}{V}\right) = \frac{k_i}{k}\,, \tag{3.58}$$

si nous avons n échantillons étiquetés $(\omega_1, \omega_2, \ldots, \omega_c)$ et k échantillons dans la région R dont k_i de la classe w_i.

En pratique, nous pouvons aussi utiliser la mesure des distances pour estimer autrement ces probabilités. Par exemple, notons $d_k^c(x)$, la distance du $k^{ème}$ plus proche voisin à x pour une classe particulière c. La probabilité estimée à l'assignation de la classe de x est :

$$\widehat{\boldsymbol{P}}_c(x) \propto \frac{1}{[d_k^c(x)]^p} \tag{3.59}$$

où p est la dimension de l'espace et n_c symbolise le nombre de points d'apprentissage de la classe c. Cette dernière mesure nous permet d'estimer une probabilité aux prédictions lors de l'utilisation de cette méthode en tant que classifieur.

3.6.3 Méthodes alternatives de classification

Bien que les méthodes de classifications présentées précédemment soient bien adaptées à la classification des régions segmentées depuis une image acquise sur une scène naturelle d'extérieur, nous décrivons brièvement deux méthodes de classification qui apparaissent très souvent dans la littérature. En effet, la première est une puissante technique relativement nouvelle qui s'appelle *Ada Boost* et la seconde traite de la classification hiérarchique.

3.6.3.1 Méthodes de *Boosting*

La pondération de la performance d'une méthode a permis de développer des modèles particuliers qui gèrent bien les situations ou cas problématiques (ou aberrants) pour les méthodes classiques. Ainsi, il existe plusieurs variantes de la méthode appelée « AdaBoost » *(Adaptive boosting)* considérées comme des techniques d'apprentissage avancé utilisées dans l'optimisation des classifieurs. Cette méthode est non seulement la plus efficace en pratique, mais également celle reposant sur des propriétés théoriques solides. La mise à jour adaptative de la distribution des échantillons, visant à augmenter le poids de ceux qui ont été mal appris par le classifieur précédent, permet d'améliorer les performances des algorithmes d'apprentissage.

Adaptive Boosting peut être très efficace avec des données de dimensions élevées en gérant plusieurs classifieurs dits « débiles » de telle manière qu'ils se complètent [Athitsos 04]. Les méthodes de *boosting* sont généralement des classifieurs binaires qui doivent être adaptés pour réaliser des tâches de reconnaissance multi-classes. Cette méthode a été exploitée au LAASdans le contexte de la détection et le suivi temporel des visages [Brethes 04].

3.6.3.2 Classifieur hiérarchique

Les arbres de décision sont parmi les méthodes de classification de fouille de données les plus populaires. Cette approche séquentielle consiste à effectuer une série de test sur l'objet à classifier ; à chaque étape, le test courant impliquant un seul ou un très faible nombre d'attributs, permet de décider du prochain test à effectuer, ou permet de conclure sur la classification (feuille de l'arbre de décision).

Ils utilisent un diagramme de flux qui possède une structure arborescente, constituée de

- un noeud racine (l'objet peut appartenir à toutes les classes existantes) ,
- un noeud interne est associé à un « test » sur l'un des attributs, à un ensemble de classes dans lesquelles l'objet peut encore être affecté et à un ensemble de noeuds fils correspondant à des résultats mutuellement exclusifs du test.
- chaque arête issue d'un noeud, est étiquetée par un résultat possible pour le test effectué en ce noeud.
- les noeuds dits feuilles révèlent les résultats de la classification. Quand un tel noeud est atteint, l'objet ne peut plus appartenir qu'à une classe ; donc, ces noeuds ne possèdent pas de fils.
- la classification d'un échantillon est faite en parcourant l'arbre depuis la racine jusqu'à une feuille.

Mettre en oeuvre un arbre de décision consiste donc à choisir les différents tests à réaliser dans le processus de décision. R.Murrieta [Murrieta-Cid 98] avait proposé dans sa thèse, une méthode formelle pour décider de l'ordre des tests ; plusieurs auteurs (dont M.Ghallab) ont proposé des techniques optimales pour générer des arbres de décision. L'avantage est que l'on peut aisément introduire des connaissances a priori dans la prise de décision.

Notons que, comme dans les autres méthodes, il faut rajouter une classe *REJET* à laquelle seront affectés les objets n'appartenant à aucune classe connue.

3.6.4 Analyse contextuelle

Notre module de classification applique donc soit SVM, soit k-PPV. Il a été efficacement appliqué pour classifier les régions, et fournir ainsi une description 2D des scènes perçues par la caméra. Néanmoins, les erreurs associées aux régions homochromatiques (faiblement texturées) du fait d'une trop grande profondeur, sont toujours présentes.

Notons qu'il est bien probable que l'on puisse corriger cela avec des étapes successives d'apprentissage supervisée ce qui donne en général des résultats locaux favorables. En dépit de cette réduction apparente d'erreurs de classification, la base de données s'en trouvera augmentée et la performance globale ainsi que la vitesse d'exécution de la reconnaissance seront négativement affectées. En effet, si le nombre d'observations augmente suffisamment, le pouvoir discriminant de certains descripteurs (texture, couleur, ...) sera affecté et l'incertitude augmentera globalement ce qui pourrait diminuer l'efficacité des classifieurs et faire échouer le processus visuel. Ce phénomène est connu comme *sur-apprentissage*.

Pour éviter ce phénomène, il convient d'envisager l'utilisation des relations contextuelles ou d'introduire des informations 3D sur l'environnement [Kumar 03]. Ces éléments vont nous permettre d'enrichir la base d'apprentissage et d'augmenter le taux de reconnaissance. Ainsi, le contexte [Torralba 03] doit être considéré comme une riche source d'information sur l'identité d'un objet, sa position et son échelle. D'ailleurs, il a été démontré que la perception humaine utilise de manière courante l'information contextuelle lorsque les conditions de vision sont mauvaises [Serrano 04].

Par contre, en analyse de scène par ordinateur, la recherche est focalisée sur la reconnaissance d'objets pris de façon individuelle, en utilisant des critères de bas niveau sans prendre en compte leur distribution spatiale ou sémantique dans la scène. Dans des conditions favorables, l'utilisation de plusieurs indices (couleur, texture, forme, géométrie ...) suffit pour une détection et classification claire d'un objet. Cependant, dans des situations où la vision est de qualité médiocre (bruitée, lointaine, non statique, entrelacée ...) le contexte semble avoir un rôle important à jouer afin d'augmenter la fiabilité de la reconnaissance. De plus, les applications réelles nécessitent des indices sémantiques pour faciliter la navigation et la recherche d'éléments visuels.

Dans le cas où nous ne disposons pas d'évidence suffisante pour identifier certains objets, l'analyse de la structure de la scène en fonction de connaissances *a priori* sur les régularités de l'environnement, donne une source complémentaire d'information pour réaliser cette interprétation. Même quand les paramètres intrinsèques d'un objet rendent possible sa classification, l'analyse contextuelle et structurelle peut simplifier cette tâche, ou peut permettre de vérifier l'interprétation courante. Ainsi, ont émergé des outils cherchant à intégrer dans le processus de vision des connaissances externes, relatives au domaine applicatif pour lequel les images sont traitées.

3.6.4.1 Utilisation de l'information contextuelle

Certains systèmes représentent ces informaations contextuelles par des règles heuristiques exploitées pour reclassifier un échantillon mal étiqueté. [Murrieta-Cid 98] utilise la nature (classe) et la configuration des objets identifiés correctement autour d'une région dite suspecte pour lui assigner une nouvelle étiquette ; par exemple, une région classée *Ciel* en bas de l'image sera suspecte ; si elle est entourée de régions classées *Chemin* avec un degré de confiance élevé, alors, elle sera ré-affectée à la classe *Chemin*. Cette correction est dite heuristique, car il s'agit de règles de ré-écriture dont les conditions d'activation sont peu formalisées. De même, dans [Iyatomi 02], les auteurs ont développé un système à inférences floues qui exploite des informations contextuelles pour corriger les défauts de classification ; leur système est destiné à la modélisation des scènes naturelles (par segmentation couleur et texture) dont le taux maximal de reconnaissance a été d'environ 90%.

Ainsi, ces approches exploitent les attributs de l'objet et ceux de son voisinage. Néanmoins, l'emploi d'un tel ensemble de règles contextuelles va rendre plus difficile l'introduction de nouvelles classes, et va accroître le rôle du superviseur ou de l'expert, seul habilité à donner des

règles.

Par contre, [Torralba 01, Torralba 03] propose une méthode probabiliste pour représenter des informations contextuelles. Il adopte un point de vue holistique (*i.e.*, qui prend en compte la scène globalement). La détection des objets revient à évaluer la fonction :

$$P(\vec{p}, \sigma, \vec{x}, o_n \mid \vec{v}) \simeq P_l(\vec{p}, \sigma, \vec{x}, o_n \mid \vec{v}_{B(\vec{x},\epsilon)}) \,, \tag{3.60}$$

où cette densité de probabilité modélise la présence d'un objet o_n sur une position spatiale $\vec{x}$, avec une pose $\vec{p}$ et une taille σ, sachant un ensemble de mesures sur toutes les positions spatiales de l'image $\vec{v}$. Une diminution de la complexité de calcul inhérente à $\vec{v}$ est faite en considérant que les régions entourant l'objet ont des attributs indépendants de la présence de l'objet. Cela est exprimé par l'utilisation des attributs locaux $\vec{v}_{B(\vec{x},\epsilon)}$ pour un voisinage B de dimension ϵ, mesuré autour de la position $\vec{x}$. Nous pouvons séparer les attributs intrinsèques (couleur, texture, forme,...) et ceux liés aux informations contextuelles, en les considérant indépendants :

$$P(\vec{v} \mid \vec{p}, \sigma, \vec{x}, o_n) = P_l(\vec{v}_L \mid \vec{p}, \sigma, \vec{x}, o_n) \cdot P_c(\vec{v}_C \mid \vec{p}_C, \sigma, \vec{x}, o_n) \tag{3.61}$$

où P_l et P_c sont respectivement les fonctions associées aux attributs intrinsèques locaux et aux attributs contextuels.

Dans [Torralba 01] le théorème de Bayes est utilisé pour décomposer l'information de la fonction P_c par :

$$P_c(\vec{p}, \sigma, \vec{x}, o_n \mid \vec{v}_C) = P_p(\vec{p} \mid \sigma, \vec{x}, o_n, \vec{v}_C) P_s(\sigma \mid \vec{x}, o_n, \vec{v}_C) P_f(\vec{x} \mid o_n, \vec{v}_C) P_o(o_n \mid \vec{v}_C) \tag{3.62}$$

où $P_o(o_n \mid \vec{v}_C)$ donne la probabilité d'avoir un objet o_n connaissant un contexte $\vec{v}_C$, $P_f(\vec{x} \mid o_n, \vec{v}_C)$ donne les positions les plus probables où localiser l'objet o_n, $P_s(\sigma \mid \vec{x}, o_n, \vec{v}_C)$ exprime les échelles les plus prévisibles de l'objet o_n à des positions spatiales différentes et finalement $P_p(\vec{p} \mid \sigma, \vec{x}, o_n, \vec{v}_C)$ exprime les formes, prototypes, points de vues et poses de l'objet o_n dans le contexte $\vec{v}_C$. Évidemment, l'estimation de cet ensemble de probabilités a priori à partir d'une base de connaissances devient la partie la plus délicate de cet algorithme.

D'autres méthodes utilisent des statistiques contextuelles apprises depuis la base de données ; elles consistent à mesurer la co-occurrence (affinité) locale entre deux objets voisins. Par exemple, si Re est la région erronée, $Rc \in c_1, \ldots, c_n$ les possibles corrections à faire (classes candidates) et $R_{ctxt} \in Rm_1, \ldots, Rm_k$ les régions voisines impliquées par le contexte de R_e, il sera possible de calculer pour chaque candidat une probabilité d'être la bonne solution étant données les régions qui entourent la région erronée dans l'image.

Pour chaque candidat c_i est calculé $p(C_i \mid R_{ctxt})$, représentant la probabilité que c_i soit la bonne solution sachant que la région erronée R_e est entourée du contexte R_{ctxt}. Cette probabilité peut être aussi simplifiée en utilisant la règle de Bayes :

$$p(c_i \mid R_{ctxt}) = \frac{p(R_{ctxt} \mid c_i) \cdot p(c_i)}{p(R_{ctxt})} \,. \tag{3.63}$$

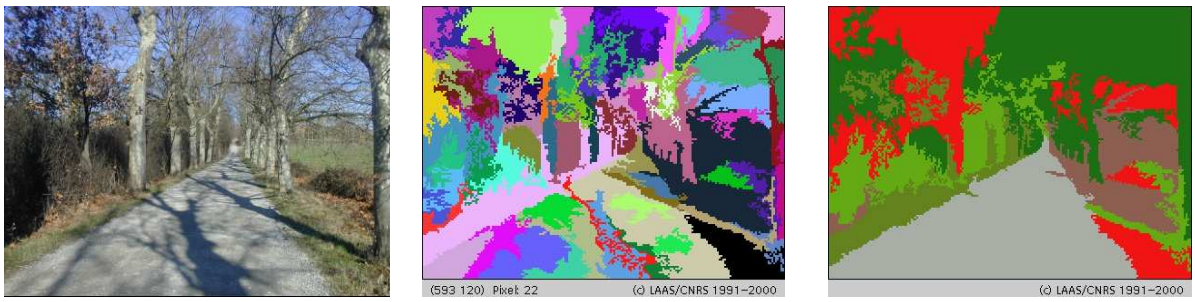
En se focalisant sur des candidats ayant une probabilité $p(R_{ctxt} \mid c_i) \cdot p(c_i)$ jugée acceptable, la probabilité $p(R_{ctxt})$ peut être considérée la même pour tous les candidats. Pour des attributs contextuels indépendants, l'approximation $p(R_{ctxt} \mid c_i) = \Pi_{j=0}^{k} p(Rm_j \mid c_i)$ permet en phase d'apprentissage, d'obtenir $p(C_i \mid R_{ctxt})$ où $P(Rm_j \mid c_i)$ symbolise le nombre de fois que Rm_j et c_i sont co-occurrents et $p(c_i)$ donne le nombre d'occurrences de c_i. Bien que cette méthode semble moins complexe que celle de Torralba, le calcul des probabilités n'est pas aisé car il nécessite une lourde phase d'apprentissage avec une quantité importante de données.

Dans notre contexte de navigation visuelle et de modélisation de l'environnement, l'information contextuelle basée sur des règles heuristiques pré-définies, destinées à corriger a posteriori des erreurs de classification, semble avoir des limitations importantes. Nous avons alors préféré utiliser des informations statistiques provenant directement du contexte de l'image. Ainsi, nous avons choisi la position spatiale (R_x, R_y) en tant qu'attributs contextuels pour caractériser une région $R_c(x, y)$ de classe c sur l'image. Ces attributs sont donnés par le centroïde normalisé de la région homogène de l'image correspondant à un objet de la classe c :

$$\vec{R}_{ctxt} = \{R_x, R_y\} = \left\{ \frac{1}{W}\frac{1}{N}\sum_{x \in Rc} x, \frac{1}{H}\frac{1}{N}\sum_{y \in Rc} y \right\}, \tag{3.64}$$

où N est le nombre de points de la région $R_c(x, y)$, W et H sont respectivement la largeur et la hauteur de l'image.

L'inclusion des attributs contextuels dans le vecteur caractérisant les régions sur l'image nous a permis d'éliminer plus efficacement plusieurs erreurs de classification (des flaques d'eau reconnues comme du ciel, des écorces et troncs d'arbre reconnus comme un chemin, des ombres d'arbre reconnues comme un arbre ...) ce qui a rendu plus robuste, la reconnaissance des objets sur des scènes naturelles. En effet, l'apprentissage de ces attributs nous donne les statistiques nécessaires pour faire des inférences à propos de la position la plus probable où identifier un certain objet.



(a) l'image d'origine (b) l'image segmentée (c) la modélisation de la scène

FIG. 3.14 – Extraction de chemin sur une image remplie d'ombres

3.6.4.2 Améliorations apportées par les informations contextuelles

La figure 3.14 illustre la puissance et l'utilité des informations contextuelles introduites dans les processus de reconnaissance. En ajoutant une information du type « position spatiale » de l'objet sur l'image, nous avons réduit considérablement le nombre de fausses détections dues à des problèmes d'ombrage et de profondeur dans les images. En effet, dans cet exemple, le chemin est correctement identifié, malgré les ombres.

Au delà de cet exemple, nous avons vérifié que l'information contextuelle s'avére très utile pour la classification de régions dans des scènes naturelles, en augmentant le taux de reconnaissance mais surtout en réduisant le nombre de fausses détections. Cela a eu des répercutions positives dans toute la chaîne de reconnaissance :

– la dimension de l'espace des attributs est augmentée, passant de $\mathbb{R}^{10}$ à $\mathbb{R}^{12}$, mais, en contrepartie, la dimension de la base de données a été réduite considérablement, passant

de plusieurs milliers d'échantillons à une centaine, grâce aux traitements par ACI et *boots-trapping*.
- la reconnaissance en utilisant k-PPV est évidemment plus rapide et une quantité moindre de mémoire a été nécessaire pour stocker les échantillons,
- l'une des améliorations les plus spectaculaires obtenue avec l'utilisation des attributs contextuels, concerne le processus d'apprentissage par la méthode des SVM . Le temps d'apprentissage en utilisant des échantillons (3500) caractérisés uniquement par la couleur et la texture dans $\mathbb{R}^{10}$ prenait généralement une journée et demi sur une station *Sun blade* 100, alors que l'information contextuelle nous a permis de faire l'apprentissage de 184 échantillons dans $\mathbb{R}^{12}$ en environ 11 s.
- enfin, une autre amélioration importante dans notre contexte : notre système est devenu plus robuste en présence d'images ombragées (*cf.* figure 3.14)

Les avantages apportés dans notre application en ajoutant des informations contextuelles sont indéniables. Il serait indispensable de prendre en compte de telles informations dans de nombreux domaines, mais, il n'est pas facile d'intégrer les connaissances du domaine dans les systèmes d'interprétation.

3.7 Résultats expérimentaux sur la description 2D des scènes naturelles

L'interprétation d'images naturelles est un domaine de recherche très actif et prometteur, tant du point de vue de la compréhension et modélisation de notre perception visuelle, que du point de vue de la recherche algorithmique en segmentation et catégorisation d'images [Celenk 95]. Elle joue un rôle important dans l'exploration et la navigation autonome des robots dans des environnements naturels inconnus. Elle permet d'extraire et de représenter de manière sémantique, les éléments principaux dans la scène ainsi que toutes les relations topologiques entre ces éléments (voisinage, inclusion. . .) dans la scène.

En robotique, la modélisation complète de l'environnement va nous donner la capacité de lancer des commandes du type : « Regarder » et « Aller vers objet », « Suivre le chemin », « Traverser le champ », *etc.* Pour ce faire, dans le chapitre 4 nous allons nous servir de ce modèle pour identifier les régions navigables.

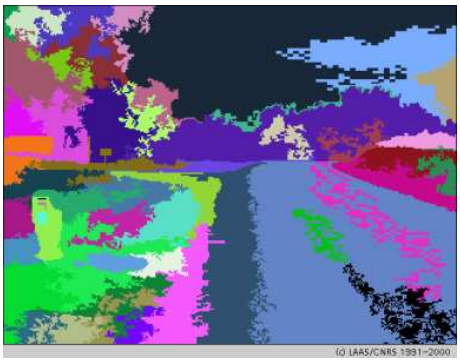
La description 2D complète de la scène courante est obtenue à partir des résultats de la classification, en fusionnant les régions homogènes appartenant à la même classe et satisfaisant un critère de connexité.

Cette fusion de régions est faite efficacement en utilisant un algorithme rapide de suivi de contours (*cf.* § 4.3.1) ; cela permet la construction d'un tableau d'adjacence qui sera exploité lors de l'opération de fusion. De plus, si la méthode de classification utilisée fournit des résultats munis d'un degré de confiance (ou probabilité d'erreur), les régions ayant une probabilité de reconnaissance élevée seront alors susceptibles d'être fusionnées. Dans le cas de la méthode k-PPV, pour vérifier la cohérence de notre approche, une mesure de confiance de l'interprétation de l'environnement, est déterminée grâce aux critères probabilistes et elle est associée à chacune des régions de l'image.

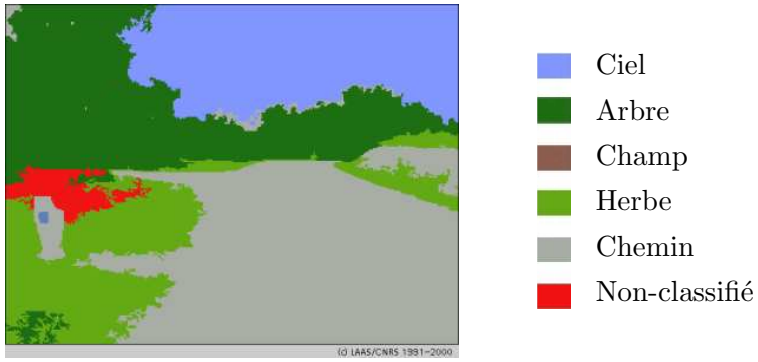
3.7.1 Résultats commentés

Les images testées ont été acquises dans différentes conditions d'illumination. Nous avons également dans notre base plusieurs types d'images, acquises avec des caméras analogiques tri-CCD,

(a) La scène d'origine



(b) L'image segmentée par couleur

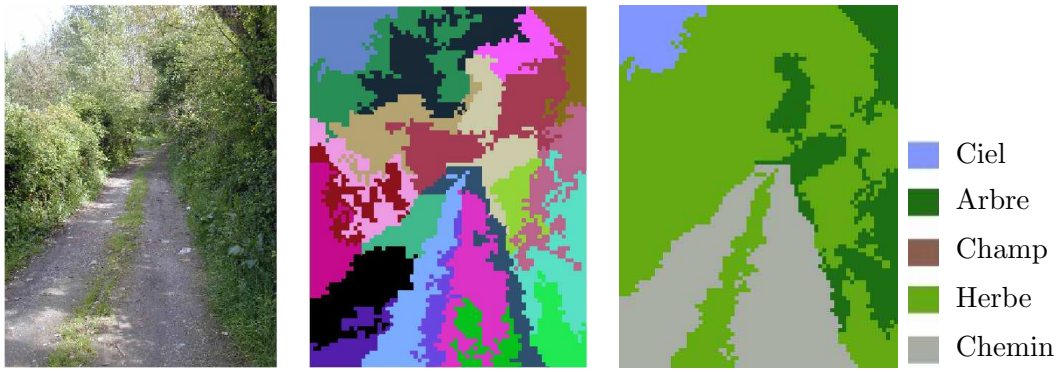


(c) Le modèle de l'environnement

FIG. 3.15 – Description 2D de la scène, obtenue avec uniquement des attributs de couleur et de texture

des caméras numériques mono-CCD, des appareils photo numériques et les caméras numériques *Micropix* C-1024 (utilisées lors de la navigation visuelle). Mis à part le cas d'images fortement ombragées, nous avons trouvé un taux d'identification très intéressant, en particulier pour les zones de type *Chemin* qui peuvent être exploitées dans le processus de navigation. Nous présentons dans les figures suivantes, plusieurs résultats, en général satisfaisants ... mais nous illustrons aussi certaines situations pour lesquelles notre méthode échoue.

- La figure 3.15 présente la description 2D d'une scène naturelle sans prendre en compte des informations contextuelles. Bien que le modèle obtenu synthétise assez bien la scène d'origine, certaines régions particulières sont encore mal classifiées : des régions lointaines, mélanges entre les classes *arbre* et *ciel*, ou régions correspondant à une classe non considérée (*e.g.*, l'objet rouge à gauche de cette image). Pour notre application à la navigation, ce n'est pas gênant car nous ne sommes pas intéressés à la reconnaissance des objets lointains ou des objets de petite dimension, mais plutôt à la détection correcte des régions adaptées à la navigation.
- la figure 3.16 présente un bon résultat, sur un chemin avec une bande centrale herbeuse, et les zones de roulement en terre. Du fait des textures très variables dans la zone de végétation, les classes *Herbe* et *Arbre* sont confondues, ce qui pourrait prêter à conséquence si le robot décidait de se déplacer sur les régions *Herbe*.



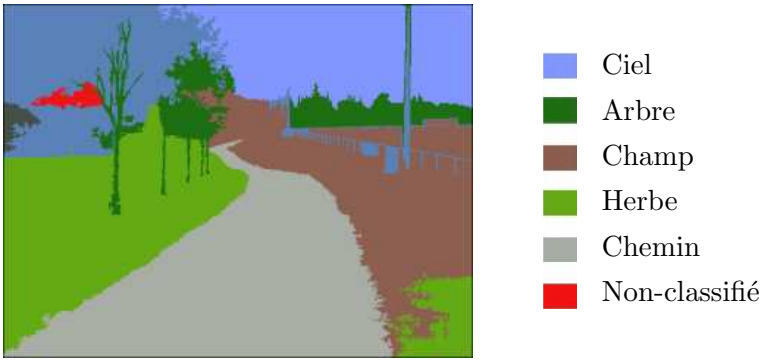
a) L'image d'origine b) l'image segmentée c) le modèle de la scène

FIG. 3.16 – Description d'une scène multi-texturée avec de forts contrastes d'illumination

- La figure 3.17 illustre un problème typique de segmentation dans les zones lointaines de l'image : les régions *Arbre* sont fusionnées avec les régions *Herbe*, mais selon les conditions, elles peuvent aussi être confondues avec *Champ*. Dans cet exemple, nous avons aussi une mauvaise segmentation des arbres : le feuillage mélangé avec le ciel doit être considèré comme une autre classe pour éviter des problèmes de fausses détections. Par ailleurs, le chemin est correctement détecté.
- Nous avions de sérieuses difficultés pour classifier les « troncs d'arbres » localisés au bord du Canal de Midi. Les régions correspondants aux troncs étaient confondues avec la classe *Chemin* ; une telle erreur serait très gênante pour la navigation du robot. Nous avons résolu cela sans autre artifice, en utilisant les deux attributs contextuels (*cf.* figure 3.18). Dans cette même image, nous détectons que la segmentation et la classification des régions correspondant à la surface de l'eau du Canal est complètement erratique, en partie du fait des reflets. Cela montre l'importance de cette phase d'interprétation, et aussi les limites de la vision monoculaire : nous pourrions mettre en danger le robot dans un environnement mal modélisé si nous n'utilisons pas d'autres traitements (autres capteurs, autres méthodes. . .).
- La figure 3.19 illustre le type d'images que nous ne sommes encore capables d'analyser correctement. La couleur de l'herbe au fond est fusionnée avec le chemin et finalement, les arbres sans feuille au fond, ne sont pas reconnus. Comme la méthode de segmentation et classification est basée en grande partie sur la couleur, et que la saturation dans cette image est faible, nous devrions alors nous tourner sur des techniques auxiliaires quand la couleur n'est pas suffisamment discriminante.
- La figure 3.20 illustre le problème typique de saturation du capteur, quand la source lumineuse est directement en face. Nous observons que malgré cet inconvénient la région navigable obtenue est acceptable de même que le modèle global de la scène.
- L'un des points faibles de notre système est constitué par les images fortement ombragées (*cf.* figure 3.21). D'abord, les zones fortement contrastées sont segmentées irrégulièrement et la méthode de reconnaissance ne peut rien faire.
- La figure 3.22 illustre une description de la scène quasiment correcte à l'exception de régions correspondants aux buissons secs, identifiées comme comme champs labourés. Par contre la région navigable est correctement détectée même en présence d'ombrages.
- La figure 3.23 présente une image complexe, où la couleur a pratiquement disparue et le capteur est à nouveau saturé. Sans doute faudrait-il exploiter la connaissance sur la

(a) La scène d'origine

(b) L'image segmentée par couleur



(c) Le modèle de l'environnement

FIG. 3.17 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

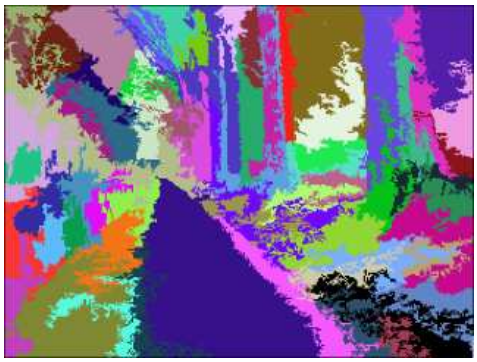
position du soleil, pour invalider un tel résultat, ou au moins, pour diminuer le degré de confiance du classifieur.

– Une image intéressante, montrant un carrefour entre un chemin de terre et une route, est présentée dans la figure 3.24. Globalement, le modèle de la scène est réussi car aucune information concernant la route (surface goudronnée) n'a été inclue dans notre base de données. Nous pouvons deviner ici, combien il sera difficile de catégoriser cette intersection !

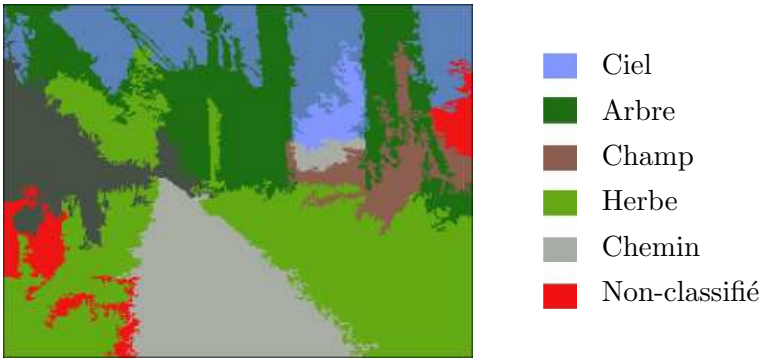
3.7.2 Évaluation globale de la méthode

Nous avons fait une première évaluation globale de notre méthode [Aviña-Cervantes 03b] en prenant 150 images pour construire la base d'apprentissage, et en gardant 30 images pour l'évaluation. Une classification supervisée des régions générées par la segmentation sur ces 30 images d'évaluation était nécessaire pour avoir des éléments de référence. Dans cette étape un opérateur humain assigne une classe à chaque région ; cette phase permet de mettre en évidence que même pour un expert, la classification peut être ambigüe. La performance de la méthode est donc évaluée en analysant cette trentaine d'images par l'algorithme et en comptabilisant le nombre de régions reconnues correctement. Dans cette première évaluation nous utilisons uniquement un vecteur de $\mathbb{R}^{10}$ caractérisé en couleur et texture. Le niveau de reconnaissance

(a) La scène d'origine



(b) L'image segmentée par couleur



(c) Le modèle de l'environnement

FIG. 3.18 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

de l'approche est en moyenne de 80%, mais il approche 90% pour les chemins : les confusions principales ont lieu entre *Arbre* et *Herbe*, mais aussi entre *Chemin* et *Champ*, surtout pour des régions lointaines où les attributs de texture sont moins représentatifs.

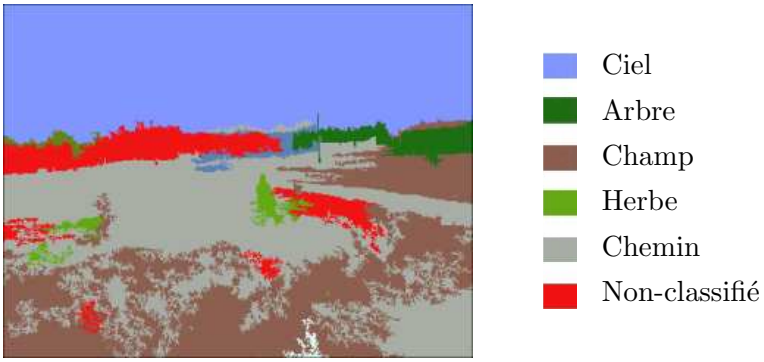
De nouvelles évaluations ont été menées ensuite pour valider l'utilisation de la méthode SVM en combinaison avec les méthodes de fouille de données d'ACP et d'ACI pour améliorer la base d'apprentissage. Ainsi, nous avons analysé à cet effet 82 images de test ; les résultats pour les classes statistiquement représentatives en classifiant avec SVM en combinaison avec une ACI sont montrés dans le tableau 3.2. En ligne figurent les résultats de reconnaissance pour les différentes classe : ainsi 7 régions *Arbre* ont été classées *Ciel*, 64 *Herbe* ...

Nous avons détecté que la méthode SVM est robuste aux variations du type de représentation des données, qui sont provoquées par un changement de base ou par une projection dans un autre espace. Ainsi, les améliorations apportées par un pré-traitement du type ACI ou ACP sur la classification par SVM ne dépassent pas 7% du taux de reconnaissance initiale, au moins pour cette base de données. Par contre, le pré-traitement de données (ACI, ACP) accélère généralement l'étape d'apprentissage dans les SVM car les données sont moins bruitées et mieux conditionnées.

Nous n'avons pas reproduit ici la matrice de confusion pour la méthode k-PPV, qui est quasiment identique à celle de SVM ; comme SVM est plus rapide en phase de classification et que nous avons très fortement réduit le temps d'entraînement de cette méthode, c'est celle-ci

(a) La scène d'origine

(b) L'image segmentée par couleur



(c) Le modèle de l'environnement

FIG. 3.19 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

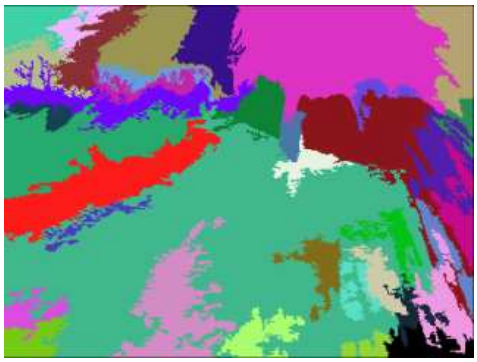
Classes	Arbre	Ciel	Herbe	Chemin	Champ	Eaux	Roches	Réussite
Arbre	613	7	64	58	3	0	32	78.89 %
Ciel	12	163	0	0	0	1	0	92.61 %
Herbe	69	0	934	8	1	2	0	92.11 %
Chemin	43	0	0	462	5	0	1	90.41 %
Champ	0	0	1	3	23	0	0	85.18 %
Eaux	2	2	1	0	0	6	0	54.55 %
Roches	11	5	0	17	0	0	41	55.41 %

TAB. 3.2 – Matrice de confusion pour évaluer la classification des régions par couleur et texture

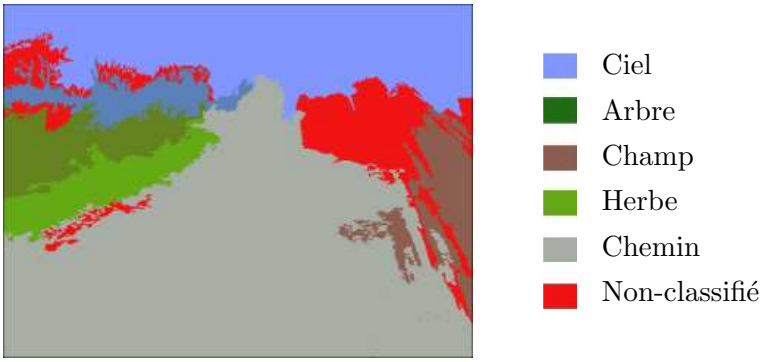
qui est exécutée à bord du robot.

Dans ces statistiques, nous avons travaillé sur des images couleur de taille de 400×300 pixels prises pendant la saison de printemps. L'algorithme de segmentation couleur prend moins de 100 ms et le temps d'exécution global jusqu'à l'obtention de la description 2D de l'image est de moins de 1 s sur un ordinateur *Sun Blade 100*.

(a) La scène d'origine



(b) L'image segmentée par couleur



(c) Le modèle de l'environnement

FIG. 3.20 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

3.8 Conclusions

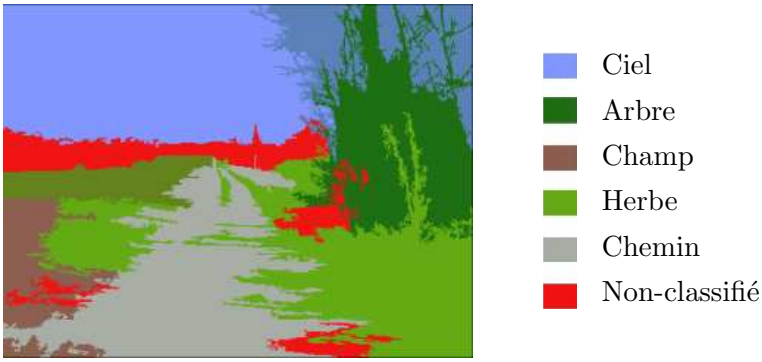
Nous avons présenté une approche pour générer une description 2D d'une scène à partir d'une simple image couleur, et d'une base d'apprentissage acquise au préalable. Cette méthode, orientée vers la détection des zones navigables, se fonde sur plusieurs opérateurs : la segmentation d'images couleur, la caractérisation et la classification des régions par la texture, la couleur et les informations contextuelles.

La technique de classification SVM a donné des résultats excellents, malgré sa convergence très lente sur notre base de données (dans la phase d'apprentissage). Cela ne nous a pas empêché de l'utiliser de façon permanente car l'étape de reconnaissance est beaucoup plus rapide et stable contre des *outliers* qu'en utilisant la méthode k-PPV. De plus, en ajoutant les paramètres contextuels nous avons détecté que la durée de la phase d'apprentissage pour les SVM a été réduite considérablement.

Cela est fondamental, car l'exploration par le robot d'un environnement inconnu a priori va impliquer des phases de ré-apprentissage périodiques. Néanmoins, il est envisageable de profiter de la technique SVM, pour accélérer l'approche k-PPV. En effet, une réduction de la complexité de la méthode k-PPV est possible en réduisant de manière pertinente le nombre d'éléments de la base de données. La technique SVM [Duda 98, Reyna-Rojas 02] sera appliquée directement sur la base de données afin de la filtrer et de conserver seulement les vecteurs les plus représentatifs.

(a) La scène d'origine

(b) L'image segmentée par couleur



(c) Le modèle de l'environnement

FIG. 3.21 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

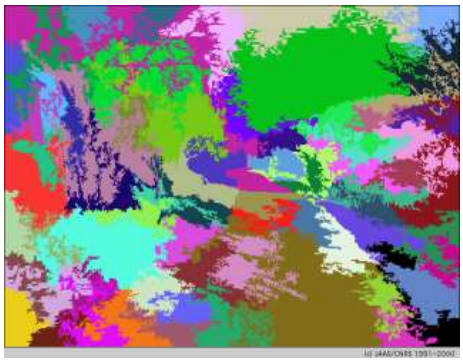
De cette manière on élimine de possibles erreurs de l'opérateur humain pendant la phase d'apprentissage (vecteurs bruités) tout en réduisant la complexité de la frontière de décision entre les classes.

La description des scènes est obtenue en moins de 1 s à partir des images couleur, avec un taux de reconnaissance de 90% pour la classe *Chemin* ; ce sont des résultats encourageants pour la suite de nos travaux de navigation. Toutefois, il sera difficile, vu la qualité des images, vu l'extrême variabilité des situations possibles dans notre contexte applicatif, d'améliorer ce résultat sans prendre en compte d'autres informations ou méthodes. Citons quelques pistes :

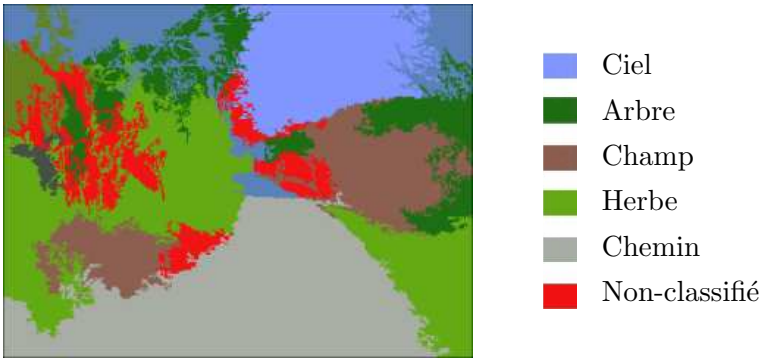
- toujours en monoculaire, prendre en compte la forme (caractérisation générique de la forme des chemins), l'orientation ou d'autres relations géométriques caractéristiques d'un chemin.
- exploiter la stéréovision pour tester la planarité des régions *Chemin*, soit en rajoutant des attributs 3D aux attributs couleur, texture et contextuels, soit en exploitant ces attributs 3D a posteriori dans une procédure de vérification.
- disposer de plusieurs bases d'apprentissage, spécialisées selon les conditions d'illumination (temps ensoleillé, couvert, pluvieux...), de saison ... Murrieta [Murrieta-Cid 02] a proposé d'extraire dans une première phase, des attributs globaux d'une image non segmentée, pour déterminer le contexte et choisir la base d'apprentissage optimale.

La détection réussie des chemins est une première étape vers la navigation de robots en milieu

(a) La scène d'origine



(b) L'image segmentée par couleur



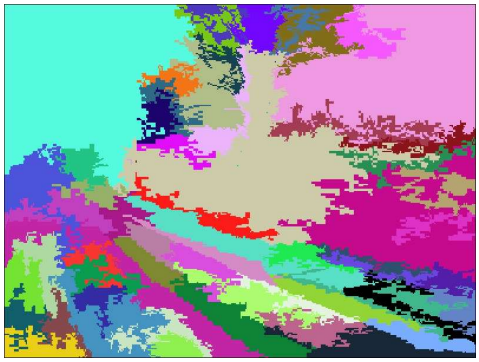
(c) Le modèle de l'environnement

FIG. 3.22 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

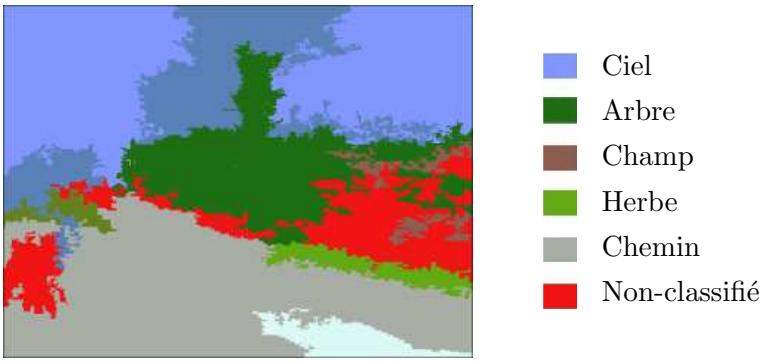
d'extérieur semi-structuré. Les contours du chemin vont nous servir pour initialiser automatiquement un algorithme de suivi temporel des bords du chemin, et pour réaliser une correction périodique de ce suivi quand cela sera nécessaire [Avina-Cervantes 03a] ; nous décrivons cette procédure dans le chapitre 5.

L'identification correcte d'autres classes que les chemins est une caractéristique intéressante de notre méthode. En effet, cette information peut être utilisée (1) pour reconnaître et localiser dans l'image, des objets d'intérêt pour guider le déplacement du robot (Aller vers un rocher, un arbre, un batiment...), (2) pour détecter des obstacles potentiels comme de grands rochers, des arbres ou des troncs d'arbres placés devant la trajectoire de déplacement, ou encore (3) pour identifier la nature du terrain, comme des zones d'eau ou de sable, navigables au sens géométrique, mais dangereuses car le robot pourrait s'y embourber.

(a) La scène d'origine



(b) L'image segmentée par couleur

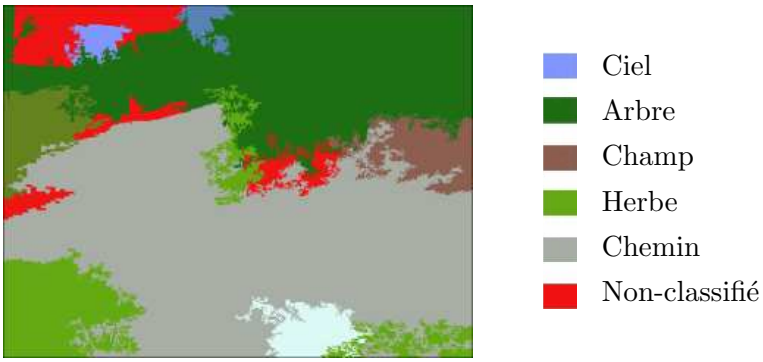


(c) Le modèle de l'environnement

FIG. 3.23 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

(a) La scène d'origine

(b) L'image segmentée par couleur



(c) Le modèle de l'environnement

FIG. 3.24 – Description 2D de la scène, en utilisant la couleur, la texture et le contexte

Chapitre 4

Modélisation topologique

4.1 Introduction

L'origine de la navigation de robots mobiles se situe dans les années 50 avec l'installation des transporteurs ou chariots guidés par fil dans les usines industrielles. Elle trouve son prolongement dans les années 70 avec le concept de suivi de routes (trajectoires fixes) où les véhicules guidés autonomes (AGVs) suivent des lignes peintes sur le sol à la place des fils enterrés sous terre (via l'inductance magnétique). Les fils enterrés étaient fiables et permanents mais ils avaient l'inconvénient de demander de considérables efforts d'installation et une subséquente inflexibilité [Tsumura 86].

Le marquage de lignes a permis de générer et de changer des trajectoires plus rapidement, mais ceci a exigé leur entretien continu pour assurer leur fiabilité (contre l'usage et l'effacement). La limitation principale de cette approche était de contraindre les déplacements sur des trajectoires fixes, ce qui limite aussi les applications potentielles des véhicules. Ainsi, le suivi de trajectoires prédéfinies a fait émerger la problématique de la localisation autonome (ou *self-localization*), puis de la navigation autonome du fait du besoin de stratégies flexibles de navigation. Le suivi de trajectoires (*Path following*) est un précurseur direct au paradigme de la localisation dans un graphe topologique, où un réseau qualitatif de chemins et de lieux est utilisé de préférence à une carte métrique.

L'essor des nouvelles technologies a permis une amélioration importante dans les processus de reconnaissance de l'environnement dans lequel le robot évolue. En effet, en dotant les robots de capteurs passifs (caméras, radars, ...) ou actifs (télémètre laser, ultrasons, infrarouges, ...), les modalités et capacités de navigation sont plus variées et plus riches. En particulier, l'extraction automatique de routes ou de chemins à partir des images numériques a été un sujet de recherche important depuis quelques années. Mais l'applicabilité réelle de ces travaux sur la détection et la navigation autonome sur des routes ou des chemins dans un environnement opérationnel, est encore peu satisfaisante [Baumgartner 99].

L'autonomie en robotique devient indispensable dans des missions que l'homme ne peut pas superviser directement (exploration planétaire, environnements dangereux pour l'être humain,...) ou tout simplement pour l'automatisation des tâches quotidiennes dans des environnements complexes. Pour cela, le schéma classique d'un système autonome implique les quatre étapes illustrées dans la figure 4.1.

Dans ce schéma, nous avons déjà discuté de l'étape de perception et partiellement de celle d'interprétation (chapitres 2 et 3), nous allons maintenant décrire les étapes de décision et d'action.

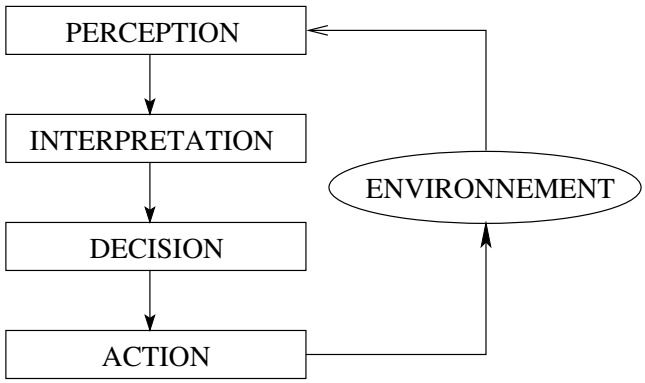


FIG. 4.1 – Schéma fonctionnel de la robotique autonome

Notre étude a porté sur l'utilisation d'une caméra CCD de manière à analyser l'environnement du robot et plus particulièrement à reconnaître les objets présents dans la scène, à partir d'une base de connaissances apprise au préalable. Cela nous permet de poser notre problème de navigation comme deux sous-problèmes : (1) « détecter » la région navigable courante sur la scène analysée, et (2) « mémoriser » les éléments sur celle-ci (*amers*) qui puissent servir de référence pour construire une carte topologique du terrain. Dans un réseau de chemins, les régions navigables sont celle étiquetées *Chemin* ;le robot doit reconnaître et mémoriser les intersections de chemins lui permettant (a) dans une phase d'apprentissage, de construire un modèle topologique du réseau dans lequel il va devoir se déplacer et (b) dans une phase de navigation, de planifier et exécuter une trajectoire topologique définie dans ce réseau. Pour cela, une reconnaissance efficace des intersections de chemins s'avère essentielle.

Ce chapitre est donc consacré à la construction d'un modèle topologique dans un réseau de chemins. La section 4.2 présente de manière générale, les approches topologiques et quelques travaux existants sur ce thème en milieu naturel. La section 4.3 fait le lien avec le chapitre précédent, puisqu'il indique comment le contour d'une région étiquetée *Chemin* sera extrait. La section 4.4 propose un descripteur d'un tel contour, en exploitant une technique proposée recemment dans la littérature appelée *Shape Context*. Enfin, la section 4.5 exploite ces descripteurs pour classifier le type de chemin perçu par le robot, en particulier pour extraire des intersections.

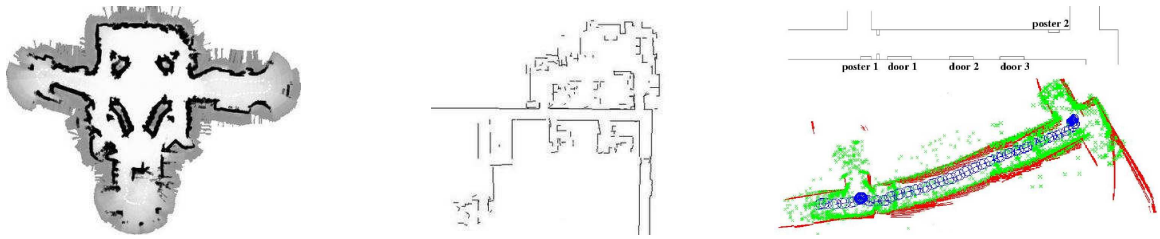
4.2 Navigation topologique en milieu naturel

4.2.1 Cartes topologiques

Pour la robotique, les cartes de navigation les plus utilisées sont les grilles d'occupation probabilistes (*occupancy grids*), les cartes d'élévation sur terrain accidenté (*digital elevation maps*), les cartes de caractéristiques (*feature maps*) et les cartes topologiques (*topological maps*). Les trois premières sont des cartes métriques où les positions des entités sont mémorisées par des coordonnées métriques, souvent uniquement 2D (x, y, θ), définies dans un système cartésien. Ces cartes métriques, qui sont à la fois les plus naturelles et les plus utilisables pour l'homme, ont fait l'objet de la majorité des travaux dans le domaine de la robotique mobile (voir figure 4.2). La construction d'une telle carte est une fonction complexe, appelée SLAM[28], qui s'exécute de manière incrémentale. Le robot se déplace dans l'environnement ; après chaque acquisition, la

[28]Simultaneous Localization and Mapping

fonction SLAM comprend plusieurs étapes essentielles : 1) l'extraction d'un modèle local (un ensemble d'observations : *features*, amers ou *landmarks*...) depuis la vue courante ; 2) l'association des observations extraites en cette position avec celles déjà mémorisées dans la carte ; 3) l'estimation de la position du robot ; 4) la fusion du modèle local dans la carte en cours de construction.



(a) Cartes par grilles d'occupation

(b) Une carte métrique

(c) Carte annotée par des amers visuels [Hayet 02]

FIG. 4.2 – Exemples de cartes métriques en milieux structurés

Au contraire, les cartes topologiques illustrent une représentation conceptuelle de l'environnement. Elles sont représentées qualitativement par une structure de graphe : les noeuds définissent des positions dans l'environnement (nommées lieux distinctifs) et les arêtes portent les commandes à exécuter pour faire des mouvements entre les noeuds qu'elles lient (*cf.* figure 4.3). Ainsi, la navigation entre deux lieux associés à deux noeuds non adjacents dans le graphe, empruntera un chemin déterminé par une séquence des commandes permettant de se déplacer entre les noeuds intermédiaires [Parra-Rodriguez 99].

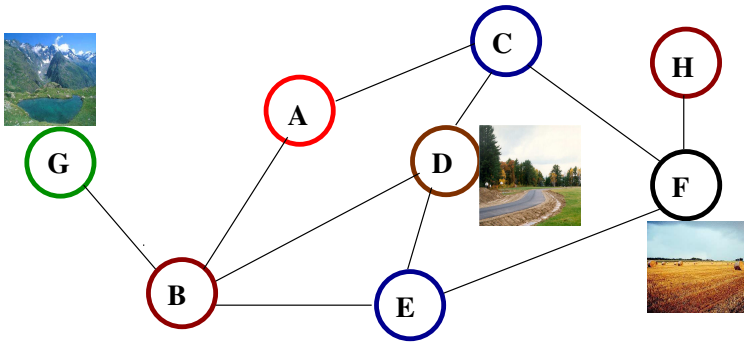


FIG. 4.3 – Carte topologique. Chaque noeud contient la description d'un lieu distinctif ; les arêtes décrivent la trajectoire reliant deux noeuds

Ce concept fonctionne sur l'hypothèse que les lieux distinctifs sont localement distinguables de leur entourage, et que l'information sur les commandes de mouvement, est suffisante pour permettre au robot de naviguer d'un noeud au suivant dans la carte ; ces commandes sont généralement référencées sur des données capteur, et non sur l'exécution d'une trajectoire explicite.

Pour la reconnaissance des lieux, la description d'un noeud doit être unique le long des trajectoires qui le relie à tous ses noeuds adjacents dans le graphe. Le robot peut par exemple reconnaître ces emplacements, en utilisant une base de données visuelles (une image associée à chaque noeud) ; le noeud est souvent un lieu spécifique de l'environnement (intersection, carrefour, passage de porte en intérieur...) ; le robot peut savoir quand il est sur un noeud ; il peut

alors acquérir une image et activer une méthode d'indexation, prenant en entrée la base des images apprises jusqu'alors sur les noeuds du graphe :

- soit l'image courante indexe une des images de la base et le robot sait en quel lieu distinctif il se trouve ; le robot dispose d'une localisation qualitative [Gonzalez-Barbosa 04].
- soit l'image courante ne correspond à aucune image apprise dans les noeuds déjà explorés du graphe : en ce cas, le robot a découvert un nouveau noeud (durant la phase de construction du graphe topologique) ou l'environnement appris au préalable a changé (par exemple, une porte fermée lors de la construction du modèle, est maintenant ouverte) ou il s'agit d'un échec de localisation.

Les cartes topologiques sont d'un grand intérêt pour traiter de la planification de trajectoires car elles représentent l'environnement de manière compacte, logique et efficace. Par exemple, il est envisageable d'élaborer des tâches du type « Suivre la trajectoire reliant noeuds A et B », ou « Aller vers noeud C ». Par nature, l'approche topologique est bien plus adaptée à une modélisation sémantique de l'espace. Enfin, les approches topologiques n'exigent habituellement pas la détermination exacte de la position métrique du robot, et de ce fait, elles supportent souvent mieux que les approches métriques, les problèmes intrinsèques aux capteurs proprioceptifs (patinage, dérives, ...).

En conséquence, la difficulté pour exploiter ces cartes topologiques est d'assurer une navigation correcte (avec identification qualitative des lieux rencontrés) sans l'utilisation directe des mesures métriques. Puisque les données sensorielles dépendent fortement du point de vue du robot, les approches topologiques ont parfois du mal à identifier les lieux qui sont géographiquement voisins, même dans des environnements statiques ; ceci rend difficile la construction de cartes de grande échelle. C'est donc la reconnaissance robuste des lieux (noeud) dans ces modèles qui est la phase la plus critique. En utilisant la vision, nous pouvons augmenter l'information disponible pour les noeuds (plusieurs images, texture, forme, *etc.*) pour mieux les reconnaître.

Le déplacement entre les noeuds est uniquement défini par une information purement qualitative, telle que « Suivre le mur » ; c'est souvent suffisant pour des environnements structurés statiques. Dans des environnements plus complexes et dynamiques, ce système de guidage peut conduire le robot dans la mauvaise direction.

En bref, les qualités des cartes métriques et topologiques sont complémentaires pour la navigation de robots mobiles. Les cartes hybrides (métriques/topologiques) sont essentiellement des structures topologiques, où la définition des lieux et des trajectoires sont définies par des informations métriques et qualitatives [Aycard 47, Guivant 04]. Dans ce schéma, un noeud n'est plus une entité discrète car il peut être défini par une région de dimensions et de forme très variées ou par une carte métrique locale.

4.2.2 Définition d'Amers

Le robot doit pouvoir se repérer dans son environnement pour exploiter les informations topologiques dont il dispose, et pour mener à bien sa navigation ; il est peu probable qu'il puisse naviguer correctement sans enrichir l'information contenue dans sa carte topologique par des *amers*. De telles entités distinctives de l'environnement, sont très souvent utilisés pour détromper les problèmes de navigation dans tous les contextes de notre vie (*e.g.*, navigation maritime, terrestre, *etc.*) ; cette notion d'*amer* ne s'applique que pour des objets (ou lieux) uniques et bien identifiables les uns des autres.

La stratégie mise en oeuvre dans la navigation visuelle consiste à donner au robot les moyens de reconnaître des éléments caractéristiques (*amers*) extraits dans l'environnement par une caméra embarquée sur le robot. Dans une approche topologique, ces *amers* vont correspondre à

des noeuds du graphe. La navigation dans un réseau de chemins a besoin d'*amers* lui permettant de mieux différencier les régions importantes de l'environnement (intersections de chemins, virages, bifurcations, *etc.*), soit pendant la construction de la carte d'un réseau, soit pendant son exploitation pour la navigation dans ce réseau. Nous considèrons donc les intersections de chemins comme des *amers*, ce qui va permettre de définir au robot, un déplacement de manière qualitative, par exemple par une requête du type « Aller vers la première intersection et tourner à gauche ».

Pour résumer, les éléments importants dans la description topologique de l'environnement sont : (1) l'identification et la reconnaissance des lieux distinctifs par l'utilisation des données sensorielles locales ; (2) la connaissance d'une commande pour se déplacer d'un lieu à un autre ; (3) un modèle topologique pour décrire la connectivité entre lieux et (4) d'une manière limité, certains descripteurs métriques de forme, de distance, de direction et d'orientation dans des repères locaux et globaux.

4.2.3 État de l'art

Les cartes topologiques sont utilisées plus souvent dans des milieux structurés d'intérieur [Thrun 98]. Par exemple dans le groupe RIA, [Chatila 85] établit un modèle topologique à partir d'un modèle géométrique, et identifie sémantiquement les éléments représentatifs de l'environnement comme des « salles », des « couloirs » à partir du modèle topologique ; leur approche emploie la description topologique pour représenter l'information de la carte à un degré d'abstraction élevé.

Simhon [Simhon 98] a proposé une construction d'une carte globale constituée par un ensemble de cartes locales qui contiennent l'information métrique sur l'environnement par rapport à un repère local. Chaque carte locale est considérée comme une île de fiabilité. Les relations entre ces cartes locales sont établies par une structure topologique hiérarchique.

Levitt [Levitt 87] a proposé une autre méthode qualitative pour l'exploration et la navigation, méthode basée sur l'identification d'amers visuels. Cette stratégie permet de naviguer avec réussite dans l'environnement en utilisant un modèle de mémoire des amers qui ne dépend pas d'un repère, sans carte précise et sans information métrique. Leur définition de lieu est basée sur des régions aux frontières virtuelles bien définies par des segments de droite reliant les amers entre eux. Cette méthode est particulièrement appropriée dans les environnements comportant des amers facilement observables depuis de nombreux points de vue, donc plutôt en extérieur.

L'un de principaux inconvénients de la représentation topologique est l'identification des lieux. Si le robot traverse deux régions très semblables, la carence d'information métrique va rendre difficile et peu fiable leur discrimination, ce qui ne permet pas au robot d'évaluer sa position. C'est pourquoi, la plupart des approches pratiques dans des environnement non structurés, mixent des informations métriques sur des cartes topologiques [Kuipers 91]. Comment obtenir ces informations métriques de manière fiable ?

- De nos jours, des systèmes autonomes ayant la capacité d'opérer dans des applications industrielles telles que le transport, l'exploitation minière, l'agriculture [Guivant 04]... utilisent un système d'information absolue GPS[29], ce qui résout ou facilite grandement les problèmes de localisation.
- Des applications plus complexes et plus robustes, ne pouvant s'appuyer totalement sur la localisation GPS (intérieur, mines, site urbain...) exploitent le concept de localisation et cartographie simultanée, référencé comme la fonction SLAM ; la plupart des applica-

[29] Global Positioning System

tions en temps réel sont basées sur le filtre de Kalman étendu (EKF). Les algorithmes de *SLAM* réalisent des étapes d'exploration et de parcours répétitifs de l'environnement, pour garantir la consistance globale de la carte construite.

En conclusion, la navigation autonome dans des environnements non structurés présente une quantité importante de problèmes non résolus ou avec une solution peu satisfaisante dans plusieurs domaines comme la perception, la localisation, la cartographie et le contrôle [Chatila 95b]. Dans ce travail, nous cherchons à fournir au robot mobile une stratégie de raisonnement qualitatif (basée sur les notions spatiales, sans faire appel à une description numérique ou quantitative) pour qu'il puisse naviguer et représenter son environnement de façon autonome, dans un réseau de chemins.

4.3 Extraction des chemins dans la scène

Dans nos travaux, la frontière de la région navigable (*e.g.*, chemin) a été extraite par une procédure élaborée qui prend en compte leurs propriétés de couleur et de texture (*cf.* § 3.7). D'abord, l'image d'origine est segmentée ; puis, les régions obtenues sont caractérisées et classifiées en exploitant les informations de couleur et de texture [Avina-Cervantes 03a] pour produire un modèle global de la scène.

4.3.1 Récupération des contours

Les pixels du bord du chemin sont extraits par un algorithme de traçage et suivi de contours sur l'image étiquetée, résultat de la description 2D de la scène. Nous avons utilisé l'information de voisinage des pixels qui appartiennent ou pas à la région à extraire (typiquement, la région *Chemin*) et nous suivons le bord de la frontière jusqu'à retrouver le point de départ. L'algorithme de traçage et suivi de la frontière d'une région est décrit dans le tableau 4.1. Cet algorithme peut être appliqué en considérant les hypothèses suivantes :

- la frontière de la région n'est pas connue mais les régions sur l'image ont été bien définies,
- l'image est binaire ou ses régions sont étiquetées,
- la frontière intérieure est un sous-ensemble des points de la région tandis que la frontière extérieure ne l'est pas.

Si la région en étude contient des trous, cette méthode ne sera pas capable d'obtenir de manière directe la totalité des contours de la région. En réalité, on devra calculer les points des *contours extérieurs* des trous de manière indépendante pour pouvoir avoir la connaissance globale des contours dans une région. Pour calculer les contours extérieurs de la région, il suffit d'utiliser la connexité à 4 voisins. La frontière extérieure contiendra tous les pixels parcourus qui n'appartiennent pas à la région. Généralement, quelques pixels sont testés plusieurs fois.

La frontière extérieure de la région est d'habitude employée pour établir des propriétés sur la forme comme, son périmètre, sa surface, sa compacité, *etc.* Il est évident que la seule utilisation du contour extérieur exclue les trous possibles de la région d´intérêt. La figure 4.6 illustre les résultats issus de l'extraction de contours à partir de l'image étiquetée 4.6.c.

4.3.2 Lissage de contours

Il est évident que les contours obtenus par la phase précédente sont parfois particulièrement bruités. Pour réduire les effets négatifs des contours bruités dans la phase d'extraction de la trajectoire à suivre pour le robot, nous avons implémenté une procédure d'échantillonnage et

- Algorithme : suivi du contour d'une région-

1 Trouver le pixel P_o (point de référence) le plus à gauche de la région d'intérêt (voir figure 4.4),
la variable *dir* stockera le précédent déplacement sur la frontière,
(a) $dir = 3$ avec une connexité à 4 voisins
(a) $dir = 7$ avec une connexité à 8 voisins

2 Parcourir les 3×3 pixels voisins du point courant dans le sens anti-horaire, en commençant par :
(a) en connexité à 4 voisins
$(dir + 3)$ [4]
(b) en connexité à 8 voisins
$(dir + 7)$ [8], si *dir* est pair
$(dir + 6)$ [8], si *dir* est impair
Le premier pixel trouvé avec la même étiquette que le pixel courant est le nouveau pixel de référence P_n

3 On arrête si le point d'origine est atteint $P_n = P_1$ et $P_{n-1} = P_o$. Dans le cas contraire, on revient à l'étape 2.

4 Les contours intérieurs sont donc représentés par la séquence $P_o \ldots P_{n-2}$.

TAB. 4.1 – Traçage et suivi des points de contour pour une région définie dans une image étiquetée

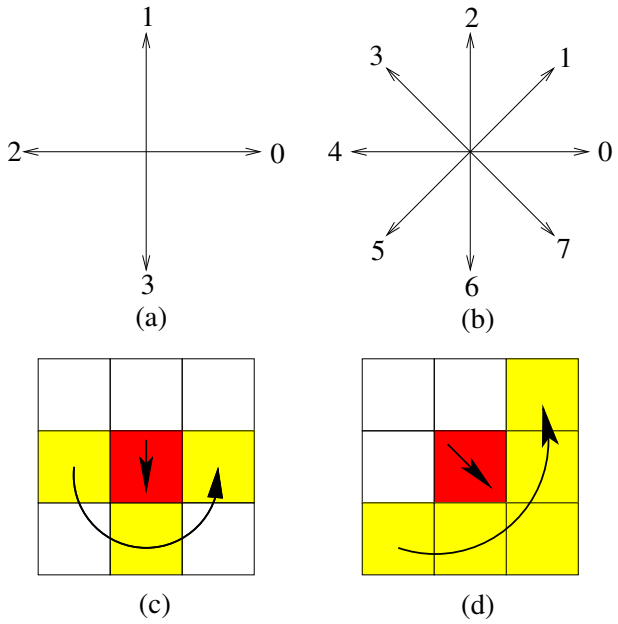


FIG. 4.4 – Suivi de pixels sur un contour. Voisinage (a) à 4 voisins, (b) à 8 voisins. Direction de rotation et séquence de recherche (c) en 4-connexité et (d) en 8-connexité.

lissage de contours. L'échantillonnage est réalisée à une fréquence variable établie arbitrairement par la sélection d'un nombre fixe de points (uniformément espacés) sur le contour.

Le lissage de contours, en particulier pour le chemin, est réalisé par une méthode d'inter-

polation utilisant les courbes de Bézier cubiques [Hu 01]. Ces courbes sont utilisées dans le but de réduire le bruit sur un contour sous-échantillonné et postérieurement lors du processus de navigation, en plaçant le robot au milieu du chemin et en calculant une trajectoire naturelle et uniforme à suivre (*cf.* § 5.4).

4.3.3 Courbes de Bézier

La courbe de Bézier est une technique rapide et efficace pour construire une trajectoire lisse et uniforme à partir d'une séquence de points. À partir de $n+1$ points de contrôle $P_0, P_1, P_2, \dots P_n$ une courbe de Bézier se définit comme :

$$C(t) = \sum_{i=0}^{n} B_{n,i}(t) P_i \tag{4.1}$$

où $t \in [0,1]$ et les coefficients $B_{n,i}$ sont des polynômes de Bernstein de la forme :

$$B_{n,i}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \tag{4.2}$$

Le point qui correspond à t est la moyenne pondérée des points de contrôle où les coefficients de pondération sont $B_{n,i}(u)$

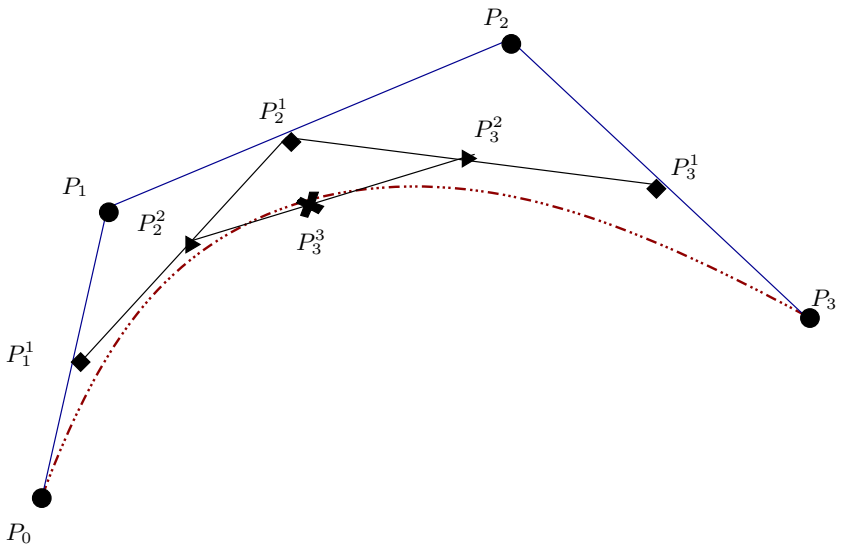


FIG. 4.5 – Courbe de Bézier

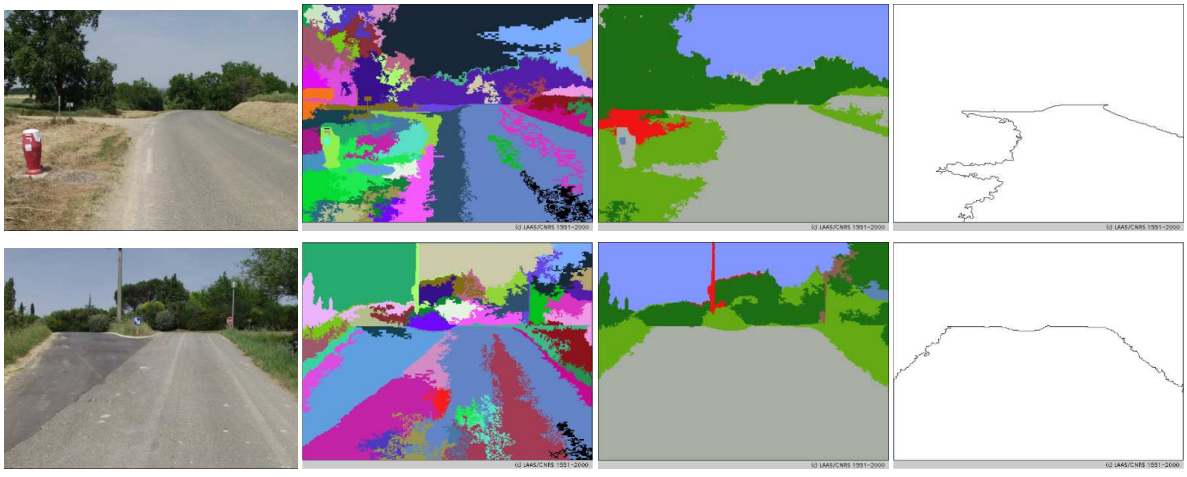
L´implémentation exploite une méthode de type « *Divide and Conquer* » illustrée dans la figure 4.5 et décrit par les équations 4.4.

$$
\begin{aligned}
P_1^1(t) &= tP_1 + (1-t)P_0 \\
P_2^1(t) &= tP_2 + (1-t)P_1 \\
P_3^1(t) &= tP_3 + (1-t)P_2 \\
P_2^2(t) &= tP_2^1(t) + (1-t)P_1^1(t) \qquad (4.3) \\
P_3^2(t) &= tP_3^1(t) + (1-t)P_2^1(t) \\
P_3^3(t) &= tP_3^2(t) + (1-t)P_2^2(t) \qquad (4.4)
\end{aligned}
$$

où $P_3^3(t)$ est le point résultat de l´opération.

4.3.4 Extraction de chemins

Les résultats de la description 2D de la scène courante doivent être directement exploitables pour la navigation visuelle. Ils fournissent l'information inhérente aux zones du terrain propices à la navigation, *i.e.*, chemins, terrains d'herbe courte.... Nous nous focalisons ainsi sur la région *Chemin*, obtenue par les étapes illustrées sur la figure 4.6. La figure présente dans la dernière colonne, les résultats finaux de l'extraction du contour de la route, définie de manière unique comme la région *Chemin* de plus grande surface, connexe au bas de l'image. Notons que, à ce stade, ces contours n'ont subi aucune opération d'échantillonnage ou de lissage ; en effet, nous travaillons avec des fréquences d'échantillonnage différentes dépendant de l'étape en cours : par exemple la méthode de catégorisation de chemins utilise généralement 200 points tandis que la méthode de *tracking* en utilise typiquement 50.



a) b) c) d)

FIG. 4.6 – Modélisation 2D de la scène et l'extraction de la route. a) Les images d'origine, b) la segmentation couleur, c) description 2D de la scène et d) l'extraction de la route (sans le lissage de contours)

Dans la section suivante, nous présentons une approche pour décrire une forme, puis pour calculer une mesure de la ressemblance entre deux formes. Cette technique sera utilisée pour la classification et la catégorisation de chemins extraits de scènes naturelles. Elle va nous permettre

d'identifier différents types d'intersections de chemins. Dans le chapitre suivant, le module de locomotion du robot se servira également de ce contour pour calculer les trajectoires.

4.4 Modélisation du chemin par la forme

Beaucoup d'applications de vision par ordinateur ont pour but la reconnaissance d'objets. Dans ces applications, la forme joue un rôle fondamental surtout quand d'autres attributs comme la couleur et la texture ne donnent pas d'information complémentaire suffisante sur la nature de l'objet [Veltkamp 01]. Par exemple, si nous parvenons à identifier (par couleur et texture) une région du type chemin sur la scène, nous ne savons rien sur sa catégorie (*i.e.*, s'il s'agit d'un chemin « tout droit », d'un « virage à gauche », d'un « virage à droite », d'une « bifurcation », d'une « intersection », *etc.*) ce qui peut être déterminée de manière naturelle en utilisant la forme. L'apparence d'un objet est en effet, largement caractérisée par des mesures de dimensions physiques basées sur la forme [Zhong 00].

Il n'y pas de définition universelle pour décrire ce qu'on appelle la « forme ». Chez l'homme, l'impression de forme est généralement accompagnée par des informations de couleur et d'intensité (texture) qui permettent de faire émerger une représentation géométrique caractéristique de l'objet [Veltkamp 99]. Ainsi, l'identification des objets est réalisée de manière courante en utilisant leurs propriétés visuelles ; nous définissons la *forme* comme un modèle géométrique constitué par un ensemble de points, courbes, surfaces solides, *etc.* .

La mise en correspondance des formes va nécessiter leur transformation dans un espace de représentation dans lequel nous pourrons les comparer, trouver leurs ressemblances, en utilisant une certaine mesure de similarité.

4.4.1 Représentations de la forme des objets

L'un des principaux problèmes pour la reconnaissance automatique des formes consiste à déterminer une représentation ou description convenable à cet effet [Zhang 04]. Elle doit être à la fois unique pour l'objet étudié et capable d'accepter certaines variations pour caractériser les éléments d'une même classe.

Les schémas de représentation de formes doivent satisfaire quelques propriétés pour donner des résultats satisfaisants : invariance à la translation, à la rotation et aux changements d'échelle. Par définition, ce type de transformations ne modifient pas la forme de l'objet et c'est plutôt en utilisant les transformations affines que l'on peut obtenir une certaine indépendance au point de vue [Flusser 92].

D'autres facteurs peuvent gêner la mise en correspondance de formes dans des scènes naturelles, par exemple les occultations et la variation dans l'éclairage de l'objet. Dans notre application, les objets de l'arrière-plan comme les arbres, les bâtiments, les voitures ou les poteaux ont une forte influence sur l'aspect (la forme) des chemins. Par conséquent, il est important de modéliser non seulement les propriétés d'un objet mais également les relations entre ceux-ci et les objets du fond de la scène.

Les différentes représentations possibles pour définir la forme des objets 2D ($O \subseteq \mathbb{R}^2$), peuvent être classifiés suivant les caractéristiques qu'elles utilisent :

- approches globales,
  - surface, périmètre, largeur, hauteur,
  - moments invariants [Belkasim 91], moments de Zernike,
  - Morphologie mathématique,
  - élongation, circularité,

- approches locales :
  - coins ou sommets (position, angles, ...),
  - segments (position, longueur, ...),
  - facettes ou régions (position, surface, couleurs, texture, ...),
- en utilisant les contours de l'objet
  - squelette,
  - représentation par des histogrammes (*Shape Context*, signature polaire)
  - descripteurs de Fourier [Pratt 96], représentation par vecteurs propres, splines...

La reconnaissance des objets 2D est menée par des méthodes différentes :

- appariement de gabarit « template matching » : les objets à détecter sont représentés par des échantillons (imagettes) ou gabarits (*templates*). L'image étudiée est balayée avec le *template* en comparant les pixels du *template* à ceux de la portion de l'image recouverte. Une mesure de comparaison du type coefficients de corrélation ou somme de différences (maximales ou absolues) sert à quantifier la similarité entre les *templates* et une portion de l'image.
- méthodes basées sur l'apparence (en exploitant l'espace propre)
- méthodes fondées sur des caractéristiques : arbres d'interprétation, alignement *etc.*

La catégorisation d'un chemin perçu dans une scène doit utiliser une représentation qui exploite le contour de la région correspondante de l'image. Le descripteur choisi pour représenter la forme d'un chemin est appelé *Shape Context*, considéré comme une alternative fiable de classification et d'identification d'objets à partir de leurs points de contour [Frome 04]. Du fait de la polyvalence et de la richesse de ce descripteur, nous l'avons adopté pour la classification et la catégorisation de chemins (de terre, goudronné...) à partir de leurs caractéristiques morphologiques (*carrefours, lignes droites, intersections en T, impasse, virages, etc.*). Notre principale motivation vise l'exploitation des intersections détectés dans un réseau de chemins, pour la construction d'une carte topologique exploitée ensuite pour la navigation autonome.

4.4.2 Catégorisation des chemins par *Shape Context*

Rappelons qu'une signature polaire d'une forme, décrite par un ensemble de points, est donnée par la distribution grossière de la forme par rapport à un point de référence ; la distribution est représentée par un histogramme polaire des points de la forme autour du point de référence. Le descripteur *Shape Context* est l'esemble des signatures polaires de la forme par rapport à chacun des points qui la décrit. La mise en correspondance entre deux formes décrites par *Shape Context*, se fait en deux étapes : (1) on rechercher pour chaque point échantillon sur une forme, le point échantillon sur l'autre qui a la signature polaire la plus semblable. (2) La maximisation de similarités globales de tous les points est résolue par l'optimisation d'un graphe bipartite [Belongie 01].

Rappelons que, pour nous, la forme est une région eextraite d'une image, et est décrite par un ensemble de points échantillonnés à partir des contours intérieurs ou extérieurs de cette région $\mathcal{P} = \{p_1, \ldots, p_n\}$, $p_i \in \mathbb{R}^2$.sur le contour de cette région de l'image (*cf.* § 4.3). Le nombre de points dépend du type d'application et de la résolution souhaitée ; pour notre application un nombre de $n = 200$ semble raisonnable. Notons qu'un lissage de contours est fort souhaitable avant l'échantillonnage des points sur ce contour.

Cette méthode n'impose aucune restriction sur les propriétés inhérentes des points représentant la forme. C'est-à-dire qu'ils ne correspondent pas à des points de courbure, ni à des maxima, des minima, des points d'inflexion, *etc.* . Pour la simplicité algorithmique, nous avons préféré travailler avec des points uniformément séparés. D'après nos expériences sur l'extraction

de contours issus des scènes naturelles, où les bords des régions sont souvent assez irréguliers, cette approche nous semble assez souple pour la représentation des chemins.

4.4.3 Descripteur « Shape Context »

Ce descripteur, pour une forme décrite par n points de contour, se compose d'un ensemble de n vecteurs ; chacun va capturer la configuration par rapport à un point quelconque du contour, de la forme décrite par les $n-1$ points restants. Cette configuration est représentée par un histogramme polaire bidimensionnel. Pour cela, il faut discrétiser la distance radiale et angulaire de tous les vecteurs, en utilisant dans ces deux grandeurs un nombre constant de *bins*. Ce nombre de *bins* dépend de la précision souhaitée et des contraintes en temps de calcul imposées par l'application. La figure 4.7 illustre l'obtention d'un histogramme, en positionnant l'axe polaire sur un point p_i du contour $\mathcal{P}$.

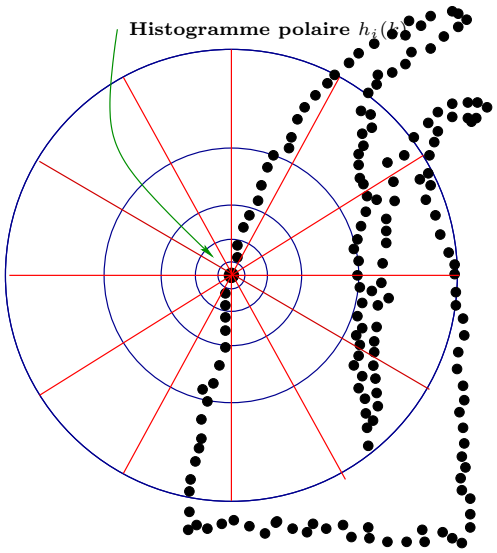


FIG. 4.7 – Calcule de l'histogramme polaire sur une point p_i

Pour un point p_i le Shape Context correspondant h_i est donné, en utilisant les coordonnées relatives des autres $n-1$ points, par :

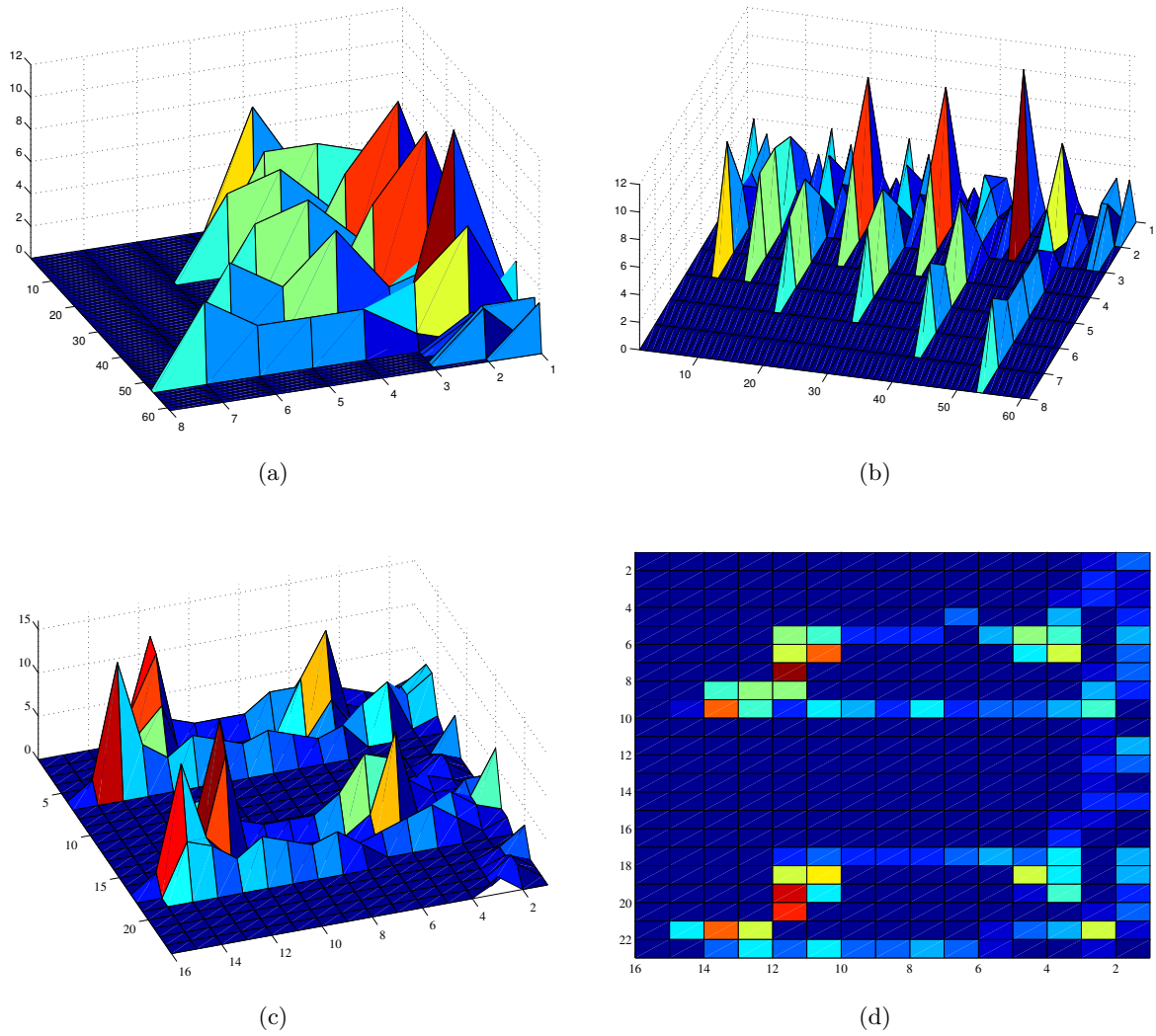
$$h_i(k) = Card\{q \neq p_i : (q - p_i) \in \text{bin}(k))\} \tag{4.5}$$

où q est un point appartenant au contour $\mathcal{P}$. Notons que dans cette expression, k dénote un *bin* sur un secteur radial dans un plan polaire ou log-polaire. La figure 4.8 présente plusieurs de ces descripteurs pour un ensemble de points échantillonnés sur le contour d'un chemin.

Les *bins* ont été choisis de manière uniforme dans l'espace *log-polaire*. Par conséquent, la sensibilité la plus grande est obtenue avec les points les plus proches. De manière pratique, un point du contour de l'objet est caractérisé par une distance radiale normalisée et un angle, de la manière suivante :

$$\theta_n = (BIN_\theta - 1)\,\frac{\theta}{2\pi} \tag{4.6}$$

$$r_n = (BIN_r - 1)\,\frac{log(1 + \frac{r}{r_o})}{log(1 + \frac{r_m}{r_o})}\,, \tag{4.7}$$



(a) (b) (c) (d)

FIG. 4.8 – Plusieurs descripteurs *Shape Context* représentant un ensemble de points échantillonnés sur le contour d'un chemin

où BIN_r et BIN_θ sont respectivement le nombre de *bins* choisis pour mesurer les rayons et les angles, r dénote la distance radiale d'un point de contour au point de référence p_i, r_m est la distance maximale de toutes les configurations et r_o dénote le rayon moyen calculé à partir des n^2 distances entre points décrivant la forme. L'utilisation de la distance moyenne r_o assure l'invariance de la méthode vis-à-vis de variations d'échelle uniformes. Par ailleurs, l'invariance à la translation est obtenue par la même définition du descripteur. Par contre, *Shape Context* n'est pas invariant aux transformations affines arbitraires mais l'utilisation de *bins* espacés de manière logarithmique assure une certaine invariance devant des petites distorsions locales. La richesse de ce descripteur est due principalement à sa robustesse aux bruits et aux occultations.

L'un des avantages inhérents à l'utilisation des histogrammes consiste à fournir une grande quantité de mesures robustes pour la comparaison, l'identification ou l'indexation des objets [Swain 91, Gonzalez-Barbosa 04]. La mise en correspondance entre deux formes distinctes consiste alors à trouver d'abord, pour chaque point p_i dans une forme, le point dans l'autre forme q_i qui

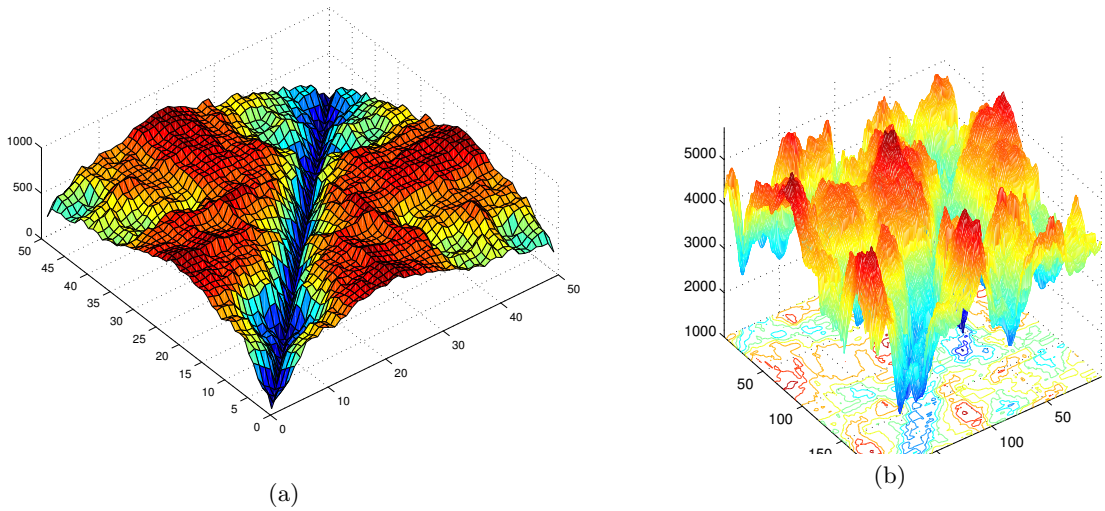
a la signature polaire la plus proche.

4.4.4 Mesure de similarité des points

Depuis quelques années, on constate une utilisation croissante des *distributions* pour représenter les attributs à mesurer au lieu de les représenter comme un vecteur dans l'espace des caractéristiques (attributs) [Rubner 98]. La représentation des formes par des histogrammes apporte des avantages pratiques, lors de leur mise en correspondance [Antani 02]. Bien qu'il y ait plusieurs mesures de similarité entre les histogrammes qui donnent des résultats satisfaisants [Gonzalez 02, Cha 02], la distance χ^2 est l'une de les plus efficaces. Ainsi, le niveau de ressemblance ($C_{ij} = C(p_i, q_j)$) entre deux points est calculé par le test statistique du χ^2 :

$$C_{ij} = C(p_i, q_j) = \frac{1}{2} \sum_{k=1} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)} \tag{4.8}$$

où C_{ij} représente le niveau de ressemblance ou le coût d'appariement entre deux points, K symbolise le nombre total de *bins*, $h_i(k)$ et $h_j(k)$ dénotent respectivement les histogrammes normalisés des points p_i et q_j. Le figure 4.9 illustre une matrice de coûts C_{ij} typique non normalisée. Celle-ci a été construite à partir de l'information fournie par deux ensembles de points (formes de 50 points).



(a) (b)

FIG. 4.9 – Distribution $2D$ non normalisée d'une matrice de coûts C_{ij}

En outre, il est possible d'enrichir le descripteur de forme en ajoutant au terme C_{ij} d'autres informations locales dérivées des positions des points p_i et q_j (*i.e.*, , l'orientation de la tangente entre les points p_i et q_j, la valeur de corrélation d'une fenêtre de pixels centrée sur ces points,. . .) ce qui peut augmenter le pouvoir discriminant de ce descripteur.

4.4.5 Mise en correspondance des formes

La métrique χ^2 nous permet de comparer deux distributions individuelles ; il est ensuite nécessaire d'appliquer une procédure permettant de mettre en correspondance deux ensembles de points car ces ensembles de points ne sont pas forcément recalés entre eux, *i.e.*, ils peuvent ne

pas avoir le même point d'origine. Une matrice contenant tous les coûts de similarité des points C_{ij} est construite à cet effet. En conséquence, pour une permutation Π entre les indices i et j des points, le coût total noté H est défini par :

$$H(\Pi) = \sum_{i=1}^{n} C(p_i, q_{\Pi(i)}) \tag{4.9}$$

Il faut donc recherche le vecteur de permutation Π qui minimise la fonction H.

Il se peut que la solution à cette équation ne soit pas unique. Pour assurer cela, la solution recherchée est sujette à la contrainte que la mise en correspondance soit bijective, ce qui est garanti en traitant notre problème comme celui de *l'assignation (quadratique) de tâches*. La méthode permettant aboutir à la mise en correspondances entre les deux formes, est basée sur l'algorithme gérant la solution d'un *graphe bipartite pondéré* [Huang 90] (BWGM, pour *bipartite weighted graph matching*), approche qui applique la théorie des graphes pour résoudre le problème de l'allocation et de l'affectation d'unités fonctionnelles. Le problème de correspondance peut être formulé comme suit :

soit le graphe dirigé G, deux listes A et B de noeuds et un tableau qui indique pour chaque arc un poids. Tous les arcs de G doivent être dirigés des noeuds A vers des noeuds B, ce qui veut dire que G doit être un graphe bipartite. Le problème consiste à trouver un ensemble bipartite de G avec la correspondance maximale, c.-à-d. un ensemble d'arcs M de telle sorte que la somme des poids de tous les arcs en M soit maximale, et que deux arcs en M ne partagent pas un même point final.

Il est alors possible de trouver une solution unique à ce problème. Jonker [Jonker 87] a développé une solution algorithmique efficace à cet effet, sa méthode reçoit une matrice carrée similaire à celle donnée par C_{ij} et fournit à la sortie le vecteur de permutation $\Pi(i)$ qui minimise l'équation 4.9.

4.4.5.1 Métrique globale

Dans notre implémentation, la métrique mesurant la ressemblance entre deux formes $\mathcal{P}$ et $\mathcal{Q}$, est fondée sur le vecteur solution du graphe bipartite pondéré $\Pi(i)$. Pour cela, nous calculons la somme symétrique de coûts qui donnent la meilleure mise en correspondance sur toutes les paires de points :

$$\mathcal{D}(\mathcal{P}, \mathcal{Q}) = \frac{1}{n} \sum_{i=1}^{n} C(i, \Pi(i)) \,, \tag{4.10}$$

où n est le nombre de points d'un contour, C est la matrice de coûts et $\Pi(i)$ le vecteur solution du graphe bipartite.

Cette mesure de similarité permet d'utiliser sans complications les *Shape Context* pour des tâches de reconnaissance ou de classification. Pour déterminer à quelle classe appartient une forme donnée, nous proposons de représenter les classes ou catégories d'objets par des prototypes, donc par des éléments idéaux au lieu d'utiliser un ensemble formel de règles logiques. L'idée des prototypes permet une appartenance douce à l'une des classes. La mise en correspondance utilisant des prototypes est généralement bien adaptée à des implémentations algorithmiques du type « plus proche voisin ». Il faudra sélectionner le nombre adéquat de vues pour chaque catégorie d'objets en fonction du temps de calcul disponible et de la précision souhaitée.

4.5 Modélisation et catégorisation de chemins

Notre objectif consiste à développer un schéma efficace pour la classification et la catégorisation de chemins. En utilisant la forme comme étape finale nous devons n'avoir que peu de sensibilité aux fortes variations d'orientation, de position et d'échelle. Nous avons conduit une série d'expérimentations pour étudier le comportement du descripteur *Shape Context* sur un ensemble de 250 images. Les histogrammes ont été construits en utilisant 16 *bins* pour la distance radiale ($log(r)$) et 24 *bins* pour l'espacement angulaire.

Notre implémentation prend environ 200 ms pour faire la mise en correspondance de deux formes en utilisant deux cent points pour les représenter. La complexité algorithmique globale dépend principalement de la solution du graphe bipartite (*i.e.*, nombre de points), laquelle est d'ordre cubique, et du nombre de *bins* dans une moindre mesure.

Si nous considérons que les deux formes sont relativement alignées (recalées) une seule signature polaire peut suffire pour représenter toute la forme. Cette signature peut être calculée, par exemple pour le centroïde de l'objet. Dans notre configuration particulière, nous considérons que le chemin est toujours en face du capteur, ce qui nous permet à la limite d'obtenir une signature sur le point :

$$P_o = P(u_o, v_o) = P\left(\min_{u \in U}(u), \frac{1}{n}\sum_{i=1}^{n} v_i\right) , \tag{4.11}$$

où $U = u_1, u_2, ..., u_n$ et $V = v_1, v_2, ..., v_n$ sont respectivement les indices ligne et colonne des points échantillonnés sur le contour. Cette supposition est basé sur le fait que le robot détecte un chemin et se positionne au milieu de celui-ci : le point de référence serait en ce cas, le point central du chemin sur la dernière ligne de l'image.

Pour compenser la diminution du nombre de descripteurs utilisés pour faire la mise en correspondance, on augmente le nombre de *bins* sur la signature polaire *Shape Context* et en même temps le nombre des points échantillonnées sur le contour. Évidemment, les formes sont convenablement normalisées pour garantir une invariance à l'échelle, translation et rotation. En conséquence, nous éviterons la complexité algorithmique inhérente au calcul de la solution d'un graphe bipartite pondéré. La mesure de similarité sera donc calculée rapidement par une distance entre les deux histogrammes représentant les signatures polaires.

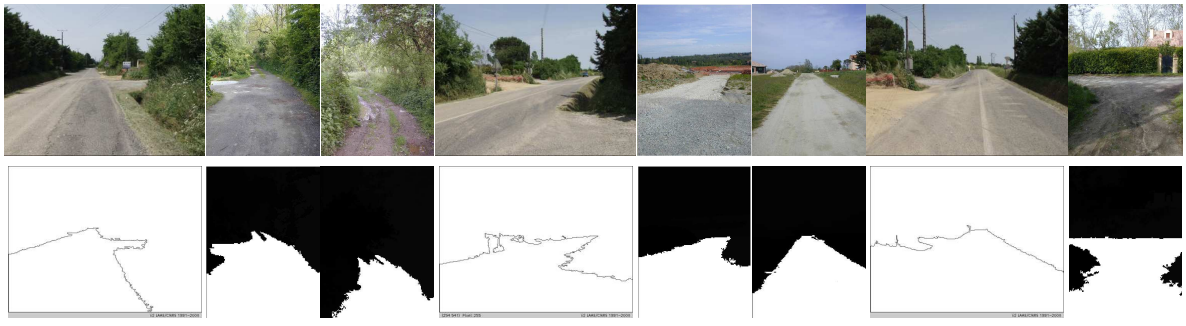


FIG. 4.10 – Quelques échantillons de la base de données de *chemins*

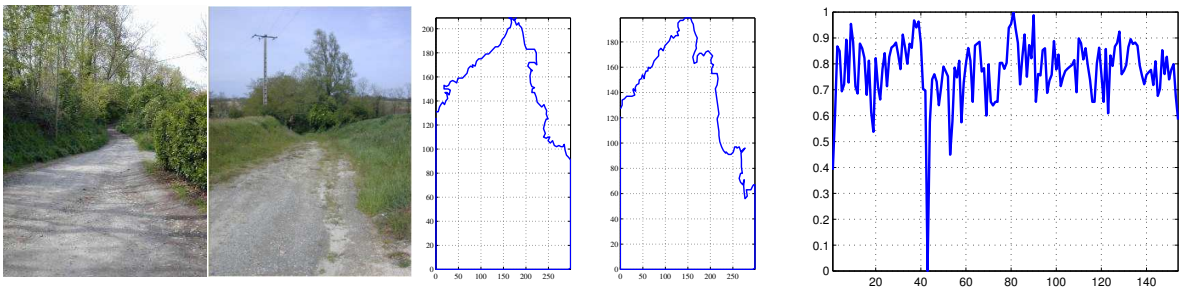
Le choix entre l'utilisation d'un descripteur *Shape Context* complet ou d'une seule signature polaire, va dépendre en pratique des conditions sur lesquelles les images sont prises ; sauf indication explicite d'exploiter seulement la signature, nous utilisons les descripteurs *Shape Context* dans leurs formes générales.

Bien que nous ayons utilisé une base d'images relativement grande, les chemins extraits ne sont pas tous adéquats (recommandables) pour les inclure dans une base de données (divisée en classes). Par exemple, il y a des chemins très larges (champ de vue réduit), d'autres ont été pris dans la mauvaise perspective, les carrefours n'étaient pas complets ..., rendant difficile de trouver des éléments représentatifs pour certains classes. Ainsi, nous avons préféré donner une étiquette à la configuration correspondante à une image de la base, puis faire une indexation sur l'ensemble des images disponibles ; l'étiquette de l'image résultat de l'indexation nous permet de reconnaître et de catégoriser les chemins. Évidemment, une base de données plus vaste et basée sur des normes bien définies lors de la prise d'images, pourrait conduire à des résultats plus satisfaisants.

4.5.1 Indexation de chemins par la forme

Pour l'indexation de chemins à partir de la forme des régions étiquetées *Chemin* dans les images, nous ne stockons que les points de contours lissés de ces régions. Les descripteurs « Shape Context » sont calculés pour toutes les images de la base de données, et stockées en mémoire pour leur réutilisation. Le but de l'indexation est de ranger ces descripteurs dans une structure qui permette de retrouver facilement les descripteurs approximativement égaux à un descripteur donné. Nous avons un critère de comparaison du type plus proche voisin entre les descripteurs de la base de données et celui de la région à identifier, *i.e.*, la forme apprise ayant la distance la plus petite (donnée par l'équation 4.10) à la forme à classifier.

La figure 4.11 montre la courbe des résultats de l'indexation du chemin perçu dans l'image « im0043 ». Ce chemin a été mis en correspondance par indexation dans une base de données composée de 156 chemins extraits d'images acquises en milieu naturel. Le descripteur « Shape Context » nous a permis d'identifier l'image « im0001 » comme la plus semblable à l'image d'entrée (*cf.* figure 4.11(b)) en utilisant les contours des chemins comme le paramètre principal. La figure 4.11(c) montre la mesure de similarité avec les 156 images de la base ; elle vaut 0 pour l'abscisse 43 (c'est la même image) et est minimale pour l'abscisse 1.

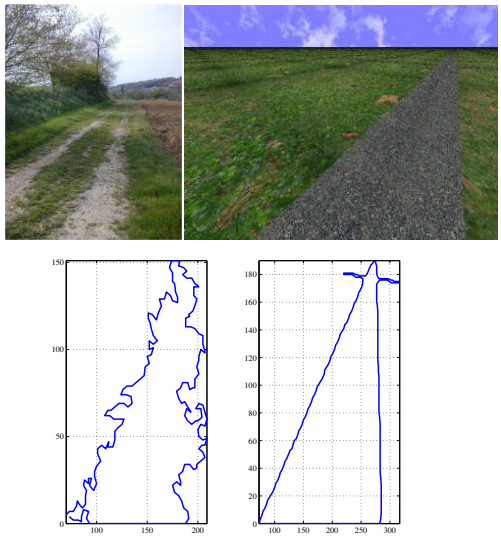


(a) Images d'origine (b) Extraction des chemins (c) Indéxation de contours

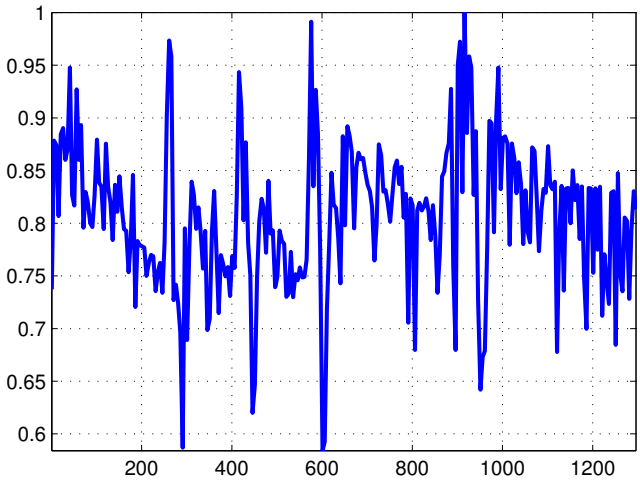
FIG. 4.11 – Indexation d'un chemin dans une base composée par des chemins extraits d'images réelles

Nous avons vérifié avec l'exemple précédent qu'il est possible d'envisager la mise en correspondance entre deux chemins extraits d'images réelles par des techniques d'indexation en utilisant le descripteur *Shape Context*. Il est possible aussi de mettre en correspondance les contours extraits l'un d'une image réelle, l'autre d'une image de synthèse comme illustré dans la figure 4.12. Le contour inconnu (à gauche) a été mis en correspondance avec 1319 images de

synthèse appartenant au réseau de chemins montré dans la figure 4.13. Le résultat obtenu est très intéressant, l'image la plus semblable est celle étiquetée *Im0600*, « virage à droite ». Évidemment, le contour extrait de l'image réelle (à gauche) représente un chemin tout droit acquis dans la mauvaise perspective (le porteur de la caméra n'est pas sur le chemin) plutôt qu'un virage à droite ; l'algorithme ne prend pas en compte cela, d'où ce résultat pas très satisfaisant. Dans un contexte de localisation, le résultat de l'indexation (en *cf.* figure 4.12(b), ensemble des mesures de similarité entre l'image réelle et les 1319 images de synthèse) nous fournirait les trois zones les plus probables où localiser cette forme de chemin : une autre technique de localisation, basée sur des amers visuels ou sur une base d'images panoramiques, serait nécessaire pour lever cette ambiguïté.



(a) Extraction des chemins



(b) Indéxation de contours

FIG. 4.12 – Indexation d'un chemin extrait depuis une image réelle sur une base composée par des contours de chemins extraits d'images de synthèse

En conclusion, nous considérons possible d'exploiter cette technique d'indexation, comme élément additionnel dans la navigation autonome et la construction de cartes topologiques. Pour ce faire, les chemins caractéristiques rencontrés dans la nature (disons : carrefours, intersection, virages, *etc.*) sont pré-classifiés en tant qu'ayant des propriétés semblables, ce qui va nous permettre d'augmenter la base de données. La catégorisation d'un chemin devient alors le processus d'indexation de cette forme dans notre base de données.

Nous avons réalisé une séquence vidéo comprenant plus d'un millier d'images de synthèse pour montrer que l'approche proposée peut aider à la construction d'une carte topologique.

4.5.2 Construction du modèle topologique

Étant donné que nous ne disposons pas pour le moment d'un terrain expérimental adéquat à ce type d'expériences, nous faisons appel à la simulation pour valider notre approche. Le réseau de chemins artificiel, sur lequel nous avons généré des images de synthèse[30] utilisant des textures réelles, est présenté en vue de dessus sur la figure 4.13.

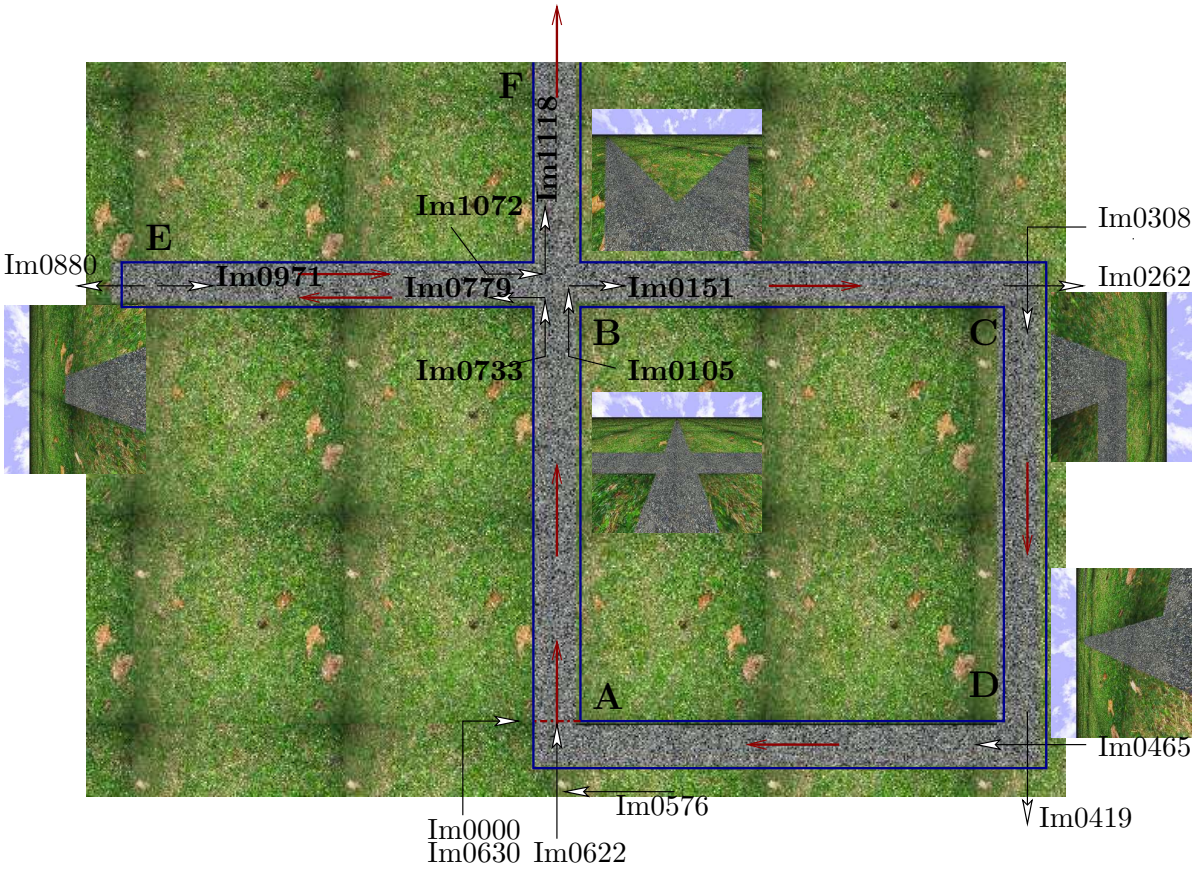
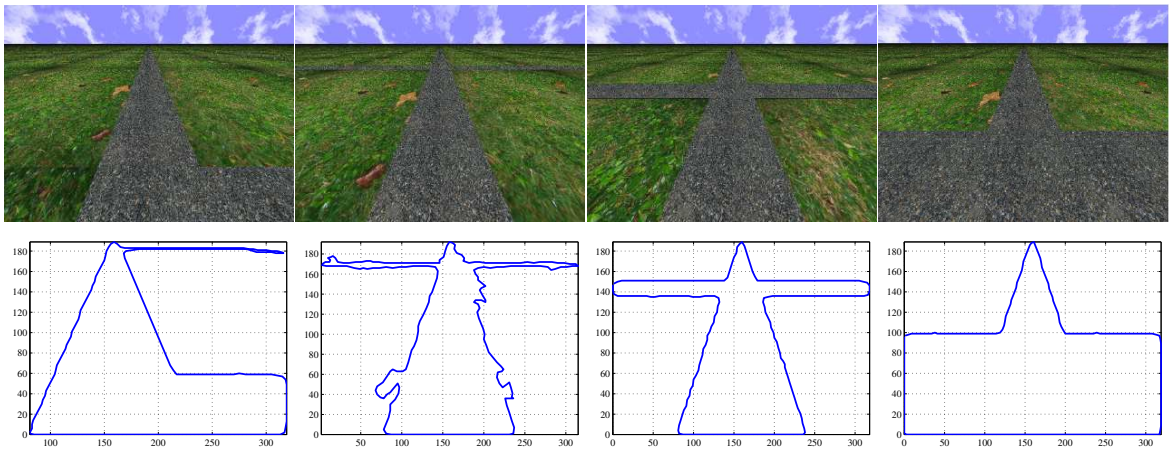


FIG. 4.13 – Vision aérienne d'un terrain expérimental de synthèse

Supposons qu'un véhicule réalise un parcours donné par la trajectoire $\mathcal{A}$ - $\mathcal{B}$ - $\mathcal{C}$ - $\mathcal{D}$ - $\mathcal{A}$ - $\mathcal{B}$ - $\mathcal{E}$ - $\mathcal{B}$ -$\mathcal{F}$. Dans une expérience réelle, cette trajectoire ne serait pas connue a priori ; le robot devrait décider en chaque carrefour de la trajectoire à suivre.

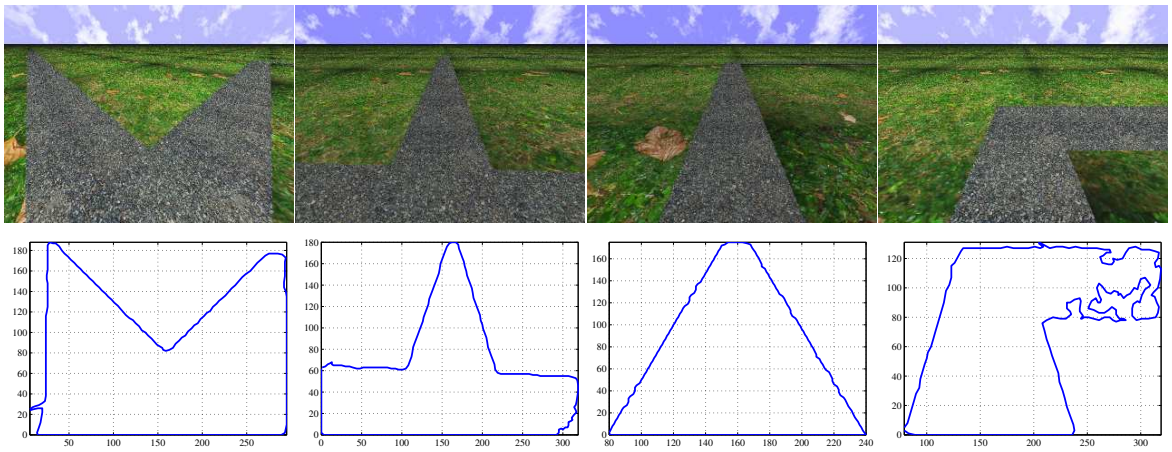
[30] Ces images ont été faites avec l'aide de V.Lemonde avec le logiciel PovRay



(a) (b) (c) (d)

FIG. 4.14 – Parcours de l'arête liant le noeud $\mathcal{A}$ et $\mathcal{B}$

Dans notre modèle topologique les *chemins droits* deviendront les arêtes de notre graphe tandis que les intersections ou bifurcations (virages, carrefours *etc.*) seront considérées comme les noeuds du graphe. Nous commençons par identifier la position de départ (figure 4.14(a)) comme un « virage à droite », qui sera donc notre noeud de départ $\mathcal{A}$. Puis nous continuons notre parcours sur un ensemble d'images (vers la droite en figure 4.14). En figure 4.14(b)) il aperçoit un carrefour lequel est clairement identifié dans la figure 4.14(c). En arrivant au carrefour (figure 4.14(d)) , nous plaçons dans notre graphe le noeud $\mathcal{B}$ relié au noeud $\mathcal{A}$ par une arête étiquetée par une commande *Suivre un chemin*.



(a) (b) (c) (d)

FIG. 4.15 – Parcours de l'arête liant le noeud $\mathcal{B}$ et $\mathcal{C}$

Sur le noeud $\mathcal{B}$, se pose le problème du choix de la direction à suivre ; nous choisissons de continuer notre parcours en tournant à droite ; les deux autres directions possibles sur le noeud $\mathcal{B}$ seront stockées dans une pile des chemins à explorer pour une analyse postérieure. Sur le noeud $\mathcal{B}$ nous tournons de 90 dégrés (figure 4.15(a) et (b)) et nous continuons le parcours sur une ligne droite (figure 4.15(c)) jusqu'à trouver une nouvelle intersection ou un virage, qui est visible sur la figure 4.15(d). Nous plaçons alors le noeud $\mathcal{C}$ sur le graphe topologique et créons une arête entre les noeuds $\mathcal{B}$ et $\mathcal{C}$.

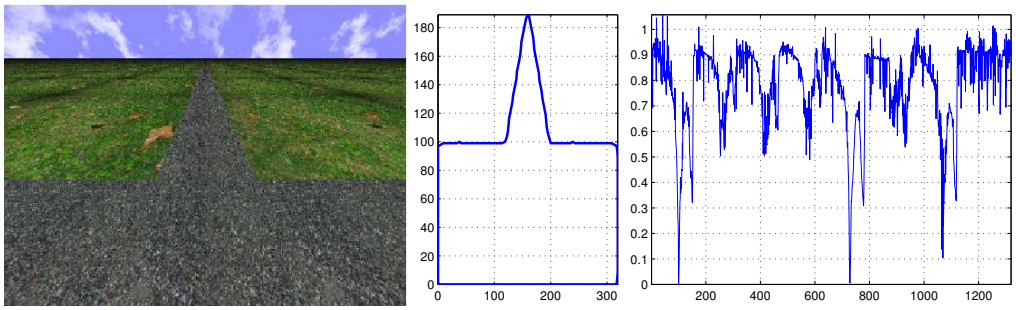
Cette procédure est exécutée jusqu'à compléter tout le graphe topologique, en vidant successivement toutes les directions à explorer dans la pile. L'un des points clefs dans la construction de ce graphe est la reconnaissance des noeuds déjà visités. En effet, n'exploiter que la forme des chemins ne sera pas suffisant dans la mesure où, avec notre approche de catégorisation, tous les carrefours en L, en T, en X... sont dans une même classe. Ainsi, l'ajout de la vision omnidirectionnelle (voir 4.16) s'avèrera d'une grande utilité pour la reconnaissance robuste des amers associés aux noeuds du graphe topologique ; à cet effet on pourra faire appel aux travaux qui ont été menés dans le groupe RIA par J. Gonzalez [Gonzalez-Barbosa 04].

FIG. 4.16 – Localisation en utilisant la vision omnidirectionelle

En prenant des images à des intervalles réguliers le long des chemins pendant une trajectoire guidée, le système extrait automatiquement la structure topologique, complétée avec la base des images panoramiques acquises en chaque noeud du graphe. Ainsi, les noeuds vont être enrichis avec une image ou un ensemble d'images s'ils appartiennent à un endroit distinctif (comme la cour d'une ferme).

Si un ensemble d'images a été acquis à fréquence régulière, pendant le parcours du robot, sur les carrefours, mais aussi pendant le suivi de chemins, le robot pourra alors répéter la trajectoire apprise en comparant l'image acquise dans sa position actuelle avec les images de référence qui ont été acquises dans la phase d'exploration, pendant le parcours de cette trajectoire. Le robot pourra alors adapter automatiquement sa position pour la mettre en correspondance avec l'image apprise la plus semblable, en se fondant sur des techniques d'asservissement visuel.

L'extraction des amers naturels, sous forme d'images panoramiques ou sous tout autre forme, est donc indispensable pour rendre plus robuste les fonctions de localisation et navigation du robot. La figure 4.17 illustre l'utilisation des « Shape Context » dans l'identification des intersections caractéristiques dans un réseau de chemins. Dans cet exemple, l'image est acquise près du noeud $\mathcal{B}$; le robot arrive au carrefour depuis le noeud $\mathcal{A}$.



(a) (b) (c)

FIG. 4.17 – L'identification d'une intersection dans un réseau de chemins

La courbe 4.17(c) montre les trois positions les plus semblables depuis lesquelles l'image courante a pu être acquise. C'est une localisation approximative qui serait déjà très fiable avant de la compléter avec l'association d'un *amer* sur l'environnement. Ainsi, l'image indexée *Im0100* (image acquise sur le noeud $\mathcal{B}$ qnand on arrive depuis le noeud $\mathcal{A}$) est très similaire à l'image *Im0725* ou l'image *Im1070* (*cf.* figure 4.13). En réalité, il est possible aussi de localiser le chemin 4.17(a) cinquante images après l'image *Im0100* comme nous pouvons le voir dans l'indexation car le robot arrive sur un carrefour symétrique.

Sans la connaissance d'une information odométrique ou un autre moyen permettant d'identifier les noeuds qu'on vient de parcourir, il serait difficile de profiter de l'information obtenue par la mise en correspondance de chemins. Il est donc indispensable de compléter cette étude avec d'autres information métriques de manière similaire au travail de J.B.Hayet [Hayet 03] dans la construction de graphes topologiques dans des couloirs en milieu intérieur.

4.6 Conclusion

Nous avons présenté une approche pour l'identification et la classification de chemins dans des scènes en milieu naturel. L'extraction et l'identification des éléments présents dans la scène sont réalisées à partir de la méthodologie décrite dans l'article [Avina-Cervantes 03a]. La description 2D de la scène permet de disposer d'une image étiquetée, depuis laquelle sont extraits les contours du chemin parcourus par le robot. La classification du chemin exploite sa forme en utilisant les propriétés du descripteur *Shape Context* de son contour [Belongie 02]. Rappelons que l'être humain emploie aussi une combinaison d'attributs (couleur, texture et forme) [Zhong 00] et exploite rarement une information d'une seule catégorie.

D'après nos expériences, nous avons trouvé que ce descripteur de forme a un grand potentiel pour la reconnaissance des objets. Les points du contour de l'objet peuvent être choisis sans restrictions importantes mais les meilleurs résultats sont obtenus avec des points uniformément séparés. Par définition la représentation des formes utilisant *Shape Context* vérifie l'invariance à la translation ; l'invariance à l'échelle est obtenue par la normalisation des vecteurs radiaux par rapport à la distance moyenne r_o. Enfin, cette représentation est robuste en présence de distorsions géométriques de petite taille (parallaxe), d'occultations et des « outliers ». Cependant, ce descripteur demande des temps de calcul (une centaine de ms) relativement considérables, surtout pour des systèmes ayant des ressources limités (mémoire, puissance du CPU,...) et qui doivent gérer plusieurs autres tâches. C'est pourquoi, à terme, nous suggérons d'utiliser cette technique à basse fréquence sur le système embarqué.

Chapitre 5

La navigation visuelle : résultats expérimentaux

5.1 Introduction

Il est tout à fait sensé de supposer que dans un environnement de type agricole (*cf.* figure 5.1), il y aura toujours un réseau de chemins permettant d'accéder aux différentes régions (entre la ferme et les champs labourés, entre différents bâtimens. . .). Le robot (machine agricole automatisée) pourra profiter des éléments structurés existants pour exécuter des primitives similaires à : suivre un chemin, longer un mur, suivre une haie, un bord de champ, une lisière . . . , traverser une région ou simplement, aller vers un objet prédéterminé.

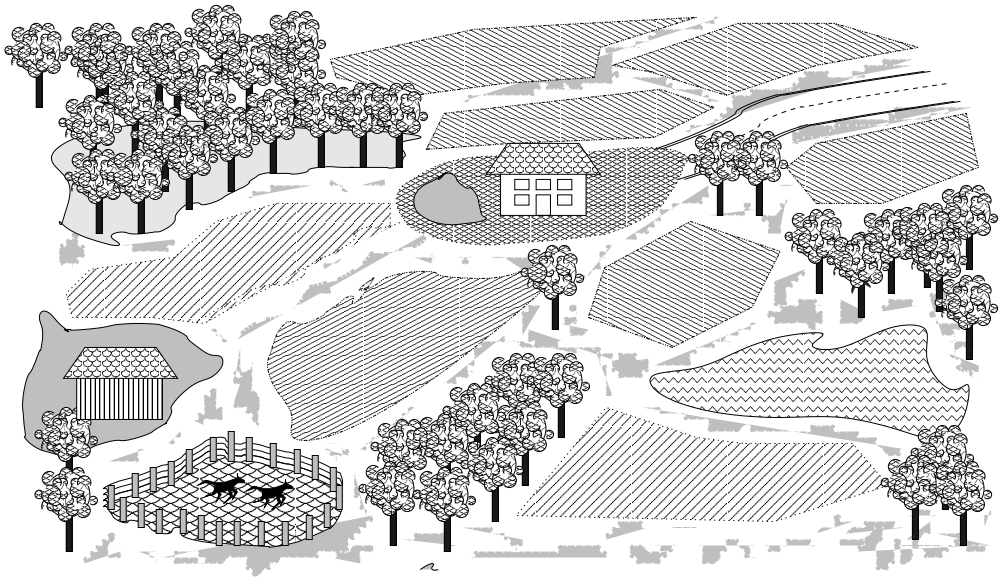


FIG. 5.1 – Environnement agricole semi-structuré

Bien que ces primitives semblent aisées, le fait de travailler dans un milieu extérieur impose une quantité importante de contraintes avant d'avoir un système fiable. Dans ces environnements, les données acquises par les ultrasons et les coupes laser sont quasiment inutilisables car les réflexions et la forme irrégulière et inconstante des objets apporteraient des informations fortement bruitées. Il est alors logique de faire appel aux caméras comme les seuls capteurs

capables de fournir des riches sources d'informations utiles au processus de navigation. Pour chacune des primitives il faudra reconnaître les cibles et régions navigables dans l'image, déduire la trajectoire du robot et finalement donner les commandes au robot et à la caméra active pour maintenir la cible dans son champ de vue. Si de plus, en phase d´initialisation, le robot doit estimer le modèle de l´environnement, il n'est pas envisageable de planifier les trajectoires avant de commencer à les parcourir. En conséquence, on a besoin d'algorithmes de traitement visuel en temps réel et de stratégies pour maintenir la cohérence des données lors de l'exécution des tâches.

Ce chapitre concerne donc l'utilisation des résultats du traitement d'images acquises en milieu extérieur, pour calculer les trajectoires et commandes du robot et de la caméra (contrôlable en *site* et *azimut*), afin d´exécuter des mouvements. Nous décrivons d'abord dans la section suivante, le mode de navigation visuelle et les primitives de mouvement qui seront exploitées dans ce mode. Puis, en section 5.3 et 5.4, nous détaillons comment les trajectoires sont calculées dans l'image, et converties en requêtes de locomotion. En section 5.5, nous décrivons comment la navigation visuelle exploite un module de suivi visuel des contours de la région d'intérêt pour la primitive en cours d'exécution. Enfin, en section 5.6, nous présentons des résultats expérimentaux obtenus sur notre démonstrateur Dala.

5.2 Primitives de mouvement

5.2.1 Modèle réactif et topologique

Un robot peut utiliser des approches réactive ou planifiée selon la façon de concevoir ses tâches. Dans un mode planifié, l'action exécutée par le robot fait partie d'un plan généré préalablement par un ensemble de planificateurs qui exploitent des représentations de l'environnement, du robot lui-même, de la mission demandée par un niveau supérieur... Au contraire, dans un mode purement réactif, un robot réalise des actions associées directement aux informations acquises par ses capteurs : par exemple, *S*i mur détecté, alors le longer, *S*i objet détecté sur le sol, alors en faire le tour ...

Pour éviter les limitations inhérentes à chacune de ces approches, une approche hybride est généralement plus robuste. Ainsi, une approche courante consiste à planifier une tâche (par exemple, une trajectoire à parcourir dans l'espace libre) et fournir au robot les capacités nécessaires pour réagir face aux événements imprévus (par exemple, un obstacle non référencé dans le modèle de l'espace libre).

Dans notre cas, le robot élabore des plans partiels qu'il suit jusqu'à ce qu'il puisse les mettre à jour en incorporant de nouvelles données provenant de ses capteurs. L'ajout de processus réactifs est envisagé comme une partie des futures améliorations, par exemple l'évitement des obstacles à l'aide de la vision stéréo : à cet effet, on fera appel aux travaux menés dans le groupe RIA par [Lemonde 04].

Afin de planifier ses déplacements, un robot mobile doit disposer d'une carte de l´environnement dans laquelle il doit exécuter des tâches. Du fait que le stockage d'une carte métrique est assez lourd et difficile à obtenir automatiquement pour des environnements extérieurs, qui plus est dynamiques, il s´avère plus approprié d'utiliser un modèle topologique : un tel modèle est un graphe dans lequel (1) les noeuds représentent les lieux significatifs de l'environnement (carrefours, amers, *etc.*), identifiables par le robot, et (2) les arêtes entre les noeuds correspondent à une primitive de mouvement permettant au robot de passer d'un lieu à un autre. Dans cet esprit, nous avons conçu une base de primitives.

A terme, un planificateur qui exploitera le graphe topologique, pourra générer des trajectoires plus complexes, exprimées comme un enchaînement entre plusieurs primitives de mouvement. De plus, un modèle hybride permettrait de mieux profiter de la structure quand elle est présente en utilisant des cartes métriques locales en certains lieux : par exemple, une carte métrique pourrait être apprise pour une cour de ferme, considéré comme un noeud dans le graphe topologique, noeud connecté à d'autres (champs, autres bâtiments) par des arêtes étiquetées par des primitives de type *S*uivre chemin.

Pendant le déroulement de chaque primitive, le robot acquiert et traite des images pour calculer une trajectoire spatiale à parcourir lors de l´itération suivante. Les morceaux de trajectoires qui n'ont pas été exécutés dans le cycle précédent seront fusionnés avec les nouvelles informations visuelles. C'est ainsi qu'une boucle de commande de supervision maintient le robot sur la bonne route en se fondant sur les indices visuels extraits à chaque itération : le calcul, la planification et l'exécution des trajectoires ont été traités en coopération avec D.Mateus [Mateus 05].

5.2.2 Modalités de locomotion

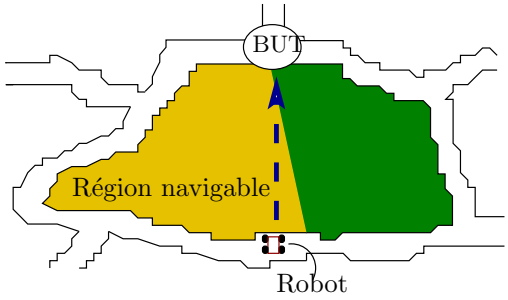
Nous avons conçu cinq primitives élémentaires de mouvement fondées uniquement sur la vision monoculaire. Le principe de fonctionnement de ces primitives est le suivant : le modèle global 2D de l'environnement va nous fournir les régions navigables ainsi que les objets à suivre [Avina-Cervantes 03a] qui sont définis dynamiquement par l'utilisateur. Les contours des régions d'intérêt pour une primitive donnée, sont représentés, pour le module de locomotion, par des courbes d'interpolation *B-splines* cubiques. Pour augmenter la vitesse à laquelle le traitement visuel décrit en chapitre 3, génére la description $2D$ de la scène et fait l'extraction de contours, une technique de suivi visuel peut être exploitée et exécutée en parallèle à la segmentation [Marin-Hernandez 04]. La figure 5.2 illustre ces primitives.

a) Suivre le chemin

b) Suivre le bord du canal

c) Aller vers l'arbre

d) Se positionner sur le chemin



e) Traverser la région

FIG. 5.2 – Primitives de mouvement en milieu extérieur semi-structuré.

Suivre un chemin Si le robot est sur un chemin, il devra extraire le milieu du contour de cette région, la trajectoire extraite sera projetée sur le sol puis décomposée en segments

de droite ou arcs de cercle. Un asservissement visuel de la platine *pan&tilt* sur laquelle est montée la caméra, permet de maintenir le chemin dans le champ de vue.

Aller vers un objet Le robot peut être commandé afin de suivre un objet reconnu dans l'image, à condition qu´il existe des régions navigables et libres d'obstacle qui relient l'objet à la position actuelle du robot. Par exemple, pour aller vers l'objet *arbre*, le robot doit traverser une région *champ*.

Suivre une bordure Le robot doit suivre une bordure ou frontière de séparation entre deux régions (dont l'une est navigable évidemment), *e.g.*, longer une haie, un mur ou une clôture.

Se positionner sur un chemin Cette primitive peut être considérée comme une initialisation d´un suivi de chemin. L´exécution de cette primitive exige l'existence d'une zone navigable permettant de rejoindre le chemin à partir de la position actuelle.

Traverser une région En fonction de la topologie du terrain, il serait utile de permettre au robot d'exécuter des raccourcis pour atteindre un but lointain plus rapidement 5.2.2. Le robot traverse une région navigable en suivant un cap, jusqu'à détecter dans l'image un but : un fossé, une haie, un chemin... Cette primitive revient à utiliser un déplacement réactif, en évitant d´éventuels obstacles jusqu'à la détection visuelle du but.

Nous n´avons pas traité dans le cadre de ce projet, l´enchaînement de ces primitives, typiquement, la séquence « Traverser la région jusqu´au chemin, Se positionner sur le chemin, Suivre le chemin ». Notons également, qu'il faudrait rajouter des primitives « Tourner à gauche », « Tourner à droite », « Traverser carrefour », dès qu'une intersection de chemin est détectée.

5.3 Calcul de la trajectoire à suivre

5.3.1 Transformations géométriques de la caméra

Deux transformations nous permettent d'associer la position d'un point de l'espace $3D$ aux coordonnées pixel de sa projection dans l'image : la première effectue la projection du point 3D sur le plan image 2D, la deuxième est une transformation affine qui change les coordonnées d'un point du plan image du repère métrique de la caméra en un repère pixel lié à l'image. Pour exprimer ces transformations, nous utilisons le modèle sténopé de la formation d'images caméra et un système de coordonnées homogènes. La figure 5.3 illustre ce modèle pour une caméra munie d'un objectif standard.

Un point P est défini en $3D$ par ses coordonnées homogènes $\boldsymbol{X}$. La projection d'un point $3D$ sur le plan image est représentée par un pixel, défini par des coordonnées sur les axes $\boldsymbol{u}$ et $\boldsymbol{v}$. Les matrices de rotation et translation seront notées respectivement $\boldsymbol{R}$ et $\boldsymbol{T}$.

Un point $3D$ de coordonnées $\boldsymbol{X}$ se projette sur le plan image R_C en un point de coordonnées $\boldsymbol{x}$, avec les relations suivantes :

$$x = f\,\frac{X}{Z} \qquad\qquad y = f\,\frac{Y}{Z} \qquad\qquad z = f\,, \tag{5.1}$$

où f est la distance focale. En coordonnées homogènes, *la projection perspective* de P est définie par l´équation matricielle suivante :

$$\begin{pmatrix} sx \\ sy \\ s \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5.2}$$

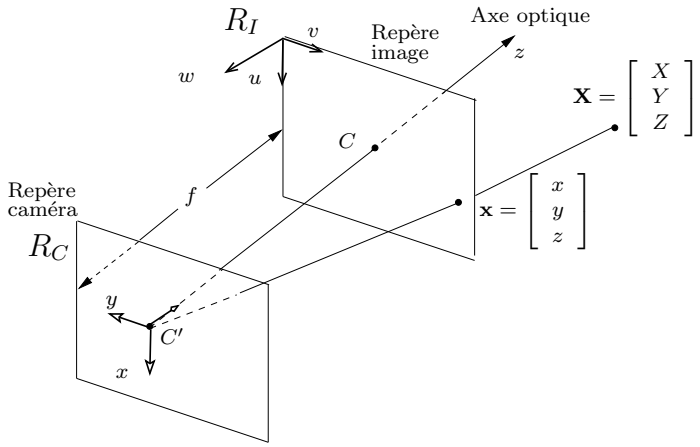


FIG. 5.3 – Modèle sténopé de la caméra

Les coordonnées pixel mesurées sur le plan image R_I dans le repère bidimensionnel u-v (voir image 5.3) sont obtenues à partir des coordonnées métriques dans le repère caméra en utilisant la transformation affine suivante :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} , \tag{5.3}$$

où u_0 et v_0 sont les coordonnées en pixels de C'. Notons que k_u et k_v sont respectivement les facteurs d'échelle vertical (pixels/m) et horizontal. L'équation 5.3 est une *transformation affine* représentant un changement d'échelle, une rotation et une translation [Gonzalez-Barbosa 04]. La relation entre les coordonnées (X, Y, Z) du point X et les coordonnées image (u, v) du point x est donnée par :

$$u = k_u\, f\, \frac{X}{Z} + u_0 \qquad\qquad v = k_v\, f\, \frac{Y}{Z} + v_0 \tag{5.4}$$

En multipliant l'équation de transformation affine 5.3 avec celle de la projection perspective 5.2 nous pouvons obtenir (à un facteur constant près) la matrice des paramètres intrinsèques I_c :

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} , \tag{5.5}$$

où les paramètres $\alpha_u = -k_u\, f$, $\alpha_v = k_v\, f$, u_0 et v_0 sont les *paramètres intrinsèques* de la caméra.

Une calibration préalable de la caméra sera nécessaire pour estimer les paramètres intrinsèques ; cela permet de calculer le rayon optique associé à un quelconque pixel de l'image, ce qui permet de déterminer, par exemple, la pose d'un objet connu par un modèle, par rapport au repère de la caméra.

Notons par ailleurs que notre caméra est embarquée sur une platine avec deux degrés de liberté.

5.3.2 Architecture de commande

Dans notre projet, une interprétation de l´image est indispensable afin de connaître les classes des régions qui entourent le robot pour faire la navigation visuelle. Les indices visuels sont définis à partir des régions d´une certaine classe et dépendent donc de la sémantique de l´environnement. Il existe de nombreux travaux sur l'*a*sservissement visuel[31], *i.e.*, sur la réalisation de mouvement référencée par la vision ; des exemples d´indices utilisés dans la littérature sont les coordonnées d´un point ou d´un ensemble de points, les paramètres d´une droite ou d´une ellipse dans le plan image, le segment (distance et orientation), la surface, le centroïde ou d´autres moments d´une région. Extraire les indices d´une telle manière est évidement coûteux en temps de calcul. En conséquence, la fréquence de l´extraction est faible (de l´ordre de 1 Hz) ce qui représente un problème du point de vue de la commande.

Dans notre contexte, il est souhaitable que les indices soient issus de régions quelconques extraites de l´image (chemin, arbre, ...) à partir d´un simple apprentissage de leurs propriétés de couleur et de texture, sans tenir compte, si possible, de la géométrie de la scène.

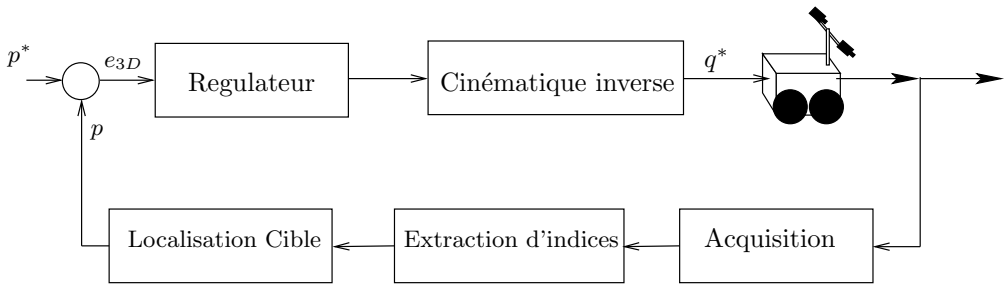


FIG. 5.4 – Configuration *position-based*. La consigne p^*, est exprimé comme la position souhaitée de la cible ou de l´effecteur dans le repère caméra

Ces raisons ont motivé le choix d'une approche d'asservissement visuel $3D$ (ou *position-based*), avec une stratégie du type *dynamic look-and-move* [Hutchinson 96, Swain-Oropeza 99]. Dans ce cas, la boucle de commande est hiérarchique, les fonctions visuelles sont utilisées pour calculer les consignes qu´un régulateur interne est chargé de stabiliser, *i.e.*, une architecture *dynamic look-and-move*. L´architecture est dite *position-based* ou *asservissement 3D* quand les indices visuels sont transformés au moyen d'un modèle géométrique pour estimer la position d'une cible $3D$ dans l'espace de travail (dans le repère caméra par exemple). La loi de commande est calculée en régulant à zéro l´erreur (dans l´espace euclidien de poses estimées) entre la position de référence de la cible vis-à-vis de la caméra et une estimation de sa position courante (voir figure 5.4). Le modèle $3D$ de la cible doit être connu, et la transformation $3D$ caméra/cible doit être estimée à chaque itération.

Notre méthode est complétée avec une technique active de suivi visuel pour maintenir la cible (chemin, arbre,...) dans le champ de vue (*cf.* figure 5.5). Cette approche de suivi visuel sera discutée dans la section 5.5.

Par ailleurs, une boucle d'asservissement en position est mise en place pour commander la platine en site et azimut. Il s'agit d'un asservissement $2D$ (ou *image-based*), puisque les consignes sont extraites directement des indices visuels dans l´image ; cette commande est censée centrer la trajectoire à suivre, en maintenant la moyenne des points de la trajectoire au centre de l'image.

[31] Le nom asservissement visuel désigne tous les systèmes qui se servent des informations extraites des images pour commander les degrés de liberté d´un système mécanique, pour nous, un robot.

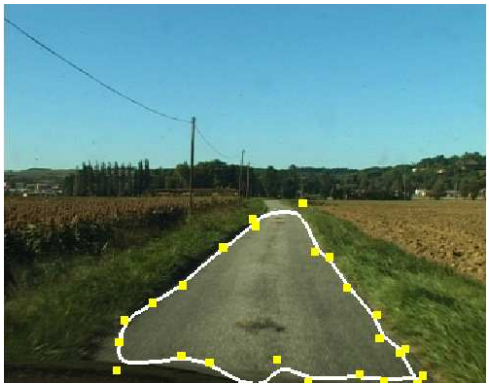


FIG. 5.5 – Application de la méthode des contours actifs pour le suivi de routes

D'autres approches pour l'asservissement de la platine, sont possibles ; on peut citer par exemple le contrôle flou utilisé lors des expérimentations de [Marin-Hernandez 04].

5.3.3 Extraction des trajectoires

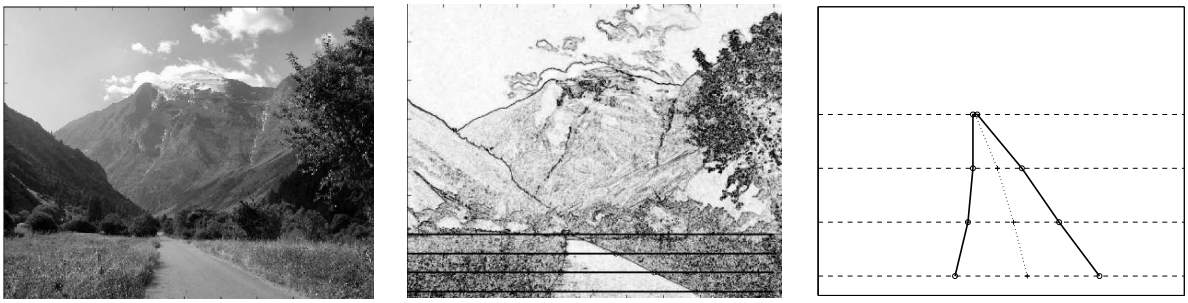
Sur chaque image, une fois le traitement visuel (*i.e.*, description $2D$ de la scène) fini, les contours des régions seront disponibles et prêts à être exploités. Selon la primitive en cours d´exécution, une région est choisie, par exemple une région *arbre* pour une primitive *Aller vers objet* ou une région *chemin* pour une primitive *Suivre un chemin*. Le traitement du contour de cette région donnera comme résultat, la trajectoire que le robot devra parcourir en quête de son but. Nous utilisons ici la technique d'extraction de contours discutée dans la section 4.3.

Les contours sont mis sous la forme de courbes polynômiales par morceaux représentées par une combinaison linéaire de *bsplines* cubiques. Cette forme compacte les rend appropriés à l'utilisation lors du suivi visuel par contours actifs *snakes* et peut être aisément calculée a partir d'une liste de points de contrôle.

Ainsi, chaque morceau entre deux points de contrôle est défini comme une paire de fonctions paramétriques $(x_i(s),y_i(s))$ et il suffit de stocker un ensemble de coefficients de deux courbes cubiques pour reconstruire le contour. Ces courbes paramétriques sont exploitées pour l'obtention de la trajectoire à suivre par le robot ; en particulier pour les primitives *Suivre un chemin* et *Suivre une bordure*.

5.3.3.1 Suivre un chemin

Maintenant, nous présentons le calcul de la trajectoire utilisée par la primitive *Suivre un chemin*. Pour extraire la trajectoire à partir du contour de la région *chemin*, les contours sont divisés par un nombre déterminé de segments horizontaux équidistants. Pour chaque segment on trouve les points de croisement (x, y) entre le contour et les droites horizontales (*cf.* figure 5.6). Cette opération implique d'abord la recherche dans le contour des intervalles qui peuvent croiser chacune des lignes horizontales. Ensuite, il faut reconstruire la courbe pour ces intervalles en récupérant x et l'abscisse curviligne s sachant que y est placé à la même hauteur que la droite croisant le contour [Mateus 05]. Quand plus de deux croisements apparaissent dans le même segment il peut s'agir d'un carrefour, d'un chemin ambigu ou bien d'une erreur dans le traitement visuel (figure 5.7). Il faut identifier et traiter proprement ces défauts de sorte que la trajectoire ne soit pas perturbée. À partir des points de croisement sélectionnés, il suffit de trouver le point



(a) Image d'origine (b) Contour et lignes horizontales (c) Schématique de l'opération

FIG. 5.6 – Extraction d'une trajectoire au milieu de la région chemin

milieu de chaque segment pour obtenir une trajectoire préliminaire.

(a) Image d'origine (b) La modélisation de la scène (c) L'extraction du chemin

FIG. 5.7 – Chemin caractérisé par une bifurcation (ambigu pour l'extraction des trajectoires)

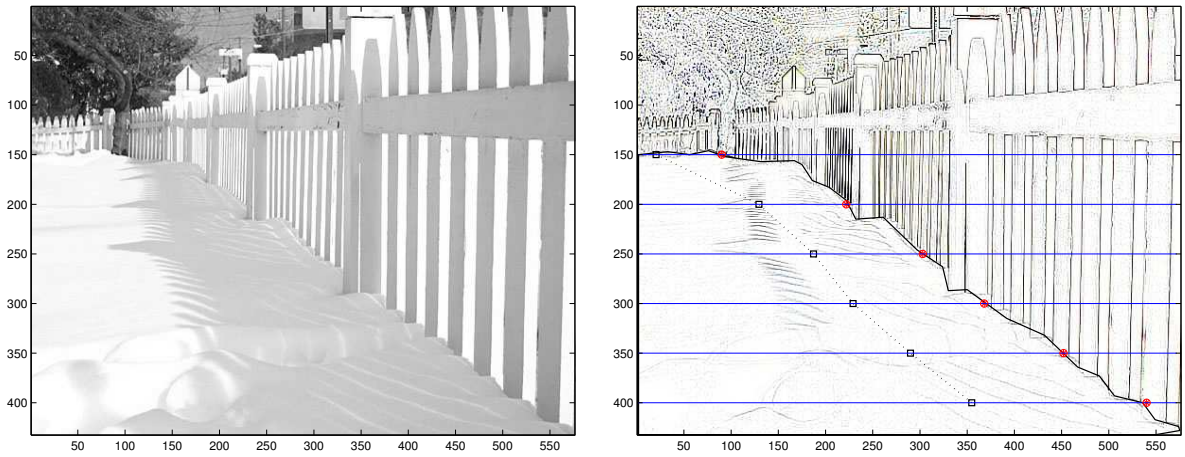
5.3.3.2 Suivre une bordure

Pour cette primitive, le point de départ est toujours le contour entre la région navigable et le contour d'une région connexe. Le robot va se guider par la bordure entre ces deux régions au lieu de trouver les points milieu de la zone navigable (voir figure 5.8).

C'est à la couche de supervision de décider quelle bordure du contour il faut suivre. La procédure est la même jusqu'au moment de trouver les points de croisement. Ensuite, la trajectoire est construite par un simple déplacement des points intersection vers l'intérieur de la région navigable ; en pratique, comme noous ne disposons pas de l'information $3D$, on projette le plan image (points de la trajectoire y compris) sur le sol en faisant l´hypothèse que le sol est une surface plane, c'est à ce moment là que le déplacement est calculé pour maintenir une distance métrique constante entre la trajectoire et la bordure.

5.3.4 Transformations géométriques : supposition de sol plat

Étant donné que le robot se déplace en suivant des trajectoires dans un monde $3D$, il faut transformer les informations issues du plan image à chaque moment, *i.e.*, projeter les points estimés dans les sections antérieures. Le calibrage de la caméra permet de connaître un rayon



(a) Image d'origine

(b) Contour, segments, points d´intersection et trajectoire

FIG. 5.8 – Extraction d´une bordure pour longer une clôture

optique, *i.e.*, la droite (*cf.* éq. 5.5) qui relie le centre optique C' de la caméra et un point quelconque $3D$ projeté sur un pixel de coordonnées (u,v).

Ces équations ne sont valables que dans le repère caméra ; nous avons préféré appliquer l'ensemble de transformations décrits ci-après pour exprimer l'équation du rayon optique dans le repère robot. En effet, nous aurons besoin des coordonnées dans le repère robot pour générer la commande.

Le calcul exact de cette profondeur avec une seule caméra impliquerait de profiter du mouvement du robot, d'appliquer une méthode de type *Motion stéréo*. Cette procédure étant trop lente, elle se révèle incompatible avec l´objectif de traitement en temps réel. C´est pourquoi nous avons plutôt choisi de faire une estimation de la distance z en supposant que le sol est plat pour les régions navigables (voir figure 5.9).

Nous préviendrons l´accumulation de l´erreur introduite par cette hypothèse (assez forte en milieu extérieur) par un calcul périodique de la trajectoire. Notons que cette hypothèse de sol plat dans un périmètre réduit autour du robot (jusqu'à 5 m devant) n'est pas très pénalisante pour le robot DALA, démonstrateur utilisé dans nos validations, qui ne peut fonctionner que sur un terrain peu accidenté. Dans un autre contexte, l'utilisation d'un *modèle numérique du terrain* de la zone devant le robot serait une solution plus adéquate.

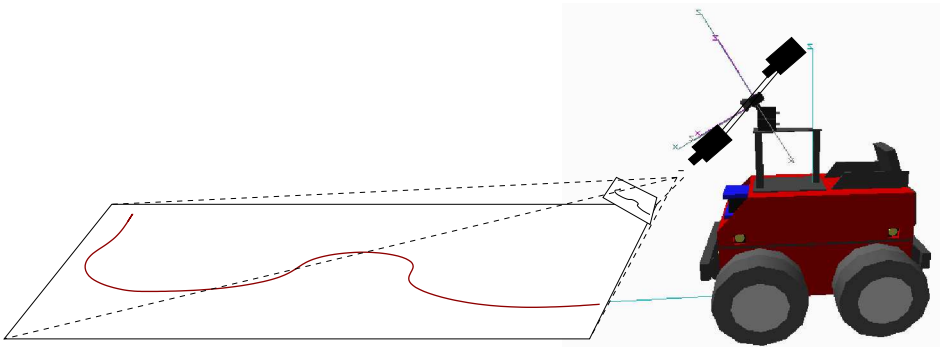
À la différence du passage de 2D à 3D, la transformation monde réel-plan image peut être complètement déterminé. Nous allons l´étudier pour faire le chemin inverse au moment de projeter la trajectoire sur le sol. Rappelons que notre caméra fait partie d'une paire stéréo montée sur une platine mobile commandable en *tilt* ϕ et en *pan* φ (figure 5.9.b).

La transformation comprend deux étapes. Dans la première, une séquence de rotations et translations permet le passage du repère principal du robot r au repère caméra c. La deuxième partie sert à transformer les coordonnées métriques du repère caméra en coordonnées pixels. D'abord, la transformation du repère caméra e au repère platine p est donnée par :

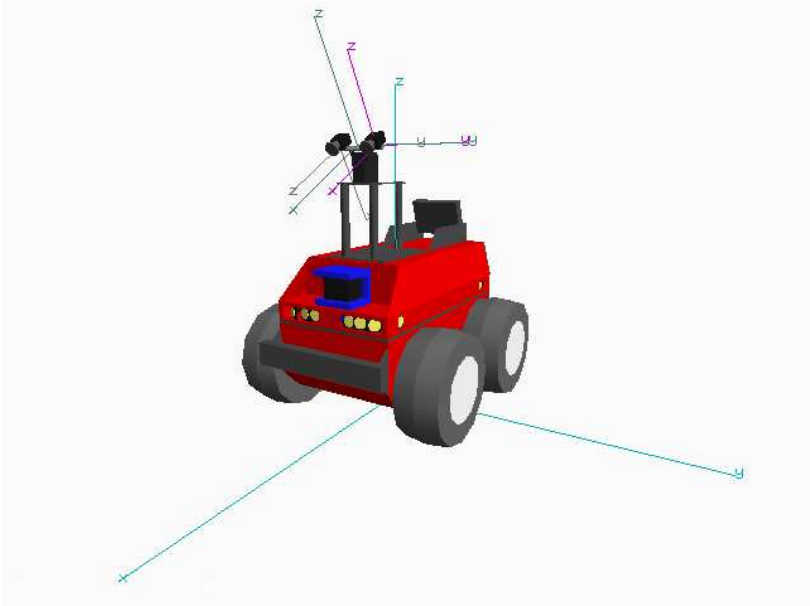
$$ {}^{p}t_{e} = \begin{pmatrix} {}^{p}_{e}t_{x} & {}^{p}_{e}t_{y} & {}^{p}_{e}t_{z} & 1 \end{pmatrix}^{t} \tag{5.6} $$

Par convention, le repère caméra c dans CALIFE[32] nécessite une rotation par rapport à celui

[32]outil pour le traitement d´images développé par le RIA-LAAS



a) Projection de la trajectoire sur le sol plat



b) Repères sur le robot DALA

FIG. 5.9 – Projection de la trajectoire sur le sol plat et les repères de DALA

qui décrit la situation de la caméra dans la paire stéréo e.

$$ {}^{e}R_{c} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} , \tag{5.7} $$

Ensuite, le passage du repère platine p au repère robot, composé de deux rotations (pan ϕ et tilt φ) et d'une translation :

$$ {}^{p}R_{r} = \begin{pmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos(\varphi) & 0 & -sin(\varphi) \\ 0 & 1 & 0 \\ sin(\varphi) & 0 & cos(\varphi) \end{pmatrix} \tag{5.8} $$

$$ \begin{pmatrix} x_r \\ y_r \\ z_r \\ 1 \end{pmatrix} = \begin{pmatrix} cos(\phi)cos(\varphi) & -sin(\phi) & -cos(\phi)sin(\varphi) & {}^{r}_{p}t_x \\ sin(\phi)cos(\varphi) & cos(\phi) & -sin(\phi)sin(\varphi) & {}^{r}_{p}t_y \\ sin(\varphi) & 0 & cos(\varphi) & {}^{r}_{p}t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_p \\ y_p \\ z_p \\ 1 \end{pmatrix} \tag{5.9} $$

Ainsi, tout l'ensemble est représenté par une matrice $\boldsymbol{T}$ comme suit :

$$\begin{pmatrix} x_r \\ y_r \\ z_r \\ 1 \end{pmatrix} = \begin{pmatrix} cos(\phi)cos(\varphi) & -sin(\phi) & -cos(\phi)sin(\varphi) & {}^{r}_{p}t_x \\ sin(\phi)cos(\varphi) & cos(\phi) & -sin(\phi)sin(\varphi) & {}^{r}_{p}t_y \\ sin(\varphi) & 0 & cos(\varphi) & {}^{r}_{p}t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & {}^{p}_{e}t_x \\ 0 & 1 & 0 & {}^{p}_{e}t_y \\ -1 & 0 & 0 & {}^{p}_{e}t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}, \quad (5.10)$$

$$\begin{pmatrix} x_r \\ y_r \\ z_r \\ 1 \end{pmatrix} = \boldsymbol{T} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix}, \qquad \boldsymbol{T} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (5.11)$$

Dans la deuxième étape de la transformation, les coordonnées pixels du point P_{uv} sont exprimées dans le repère caméra c grâce aux équations 5.5 et 5.4, nous avons la transformation complète :

$$P_{uv} = I_c\, P_c = I_c\, T^{-1}\, P_r \quad (5.12)$$

$$s \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{21} & r_{31} & -r_{11}t_1 - r_{21}t_2 - r_{31}t_3 \\ r_{12} & r_{22} & r_{32} & -r_{12}t_1 - r_{22}t_2 - r_{32}t_3 \\ r_{13} & r_{23} & r_{33} & -r_{13}t_1 - r_{23}t_2 - r_{33}t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_r \\ y_r \\ z_r \\ 1 \end{pmatrix} \quad (5.13)$$

Cette équation matricielle correspond à l'équation de deux plans dans le repère robot, donc d'une droite 3D, qui n'est autre que le rayon optique correspondant au pixel (u, v).

Pourtant, ce qui nous intéresse c'est la transformation inverse : ayant les coordonnées pixels dans l'image, comment obtenir les points correspondants dans le monde tridimensionnel (repère robot). En d'autres termes, on doit résoudre le système d'équations pour déterminer x_r, y_r et z_r à partir de la connaissance de u et v. De 5.13 nous pouvons définir les équations de deux plans :

$$a_1 x_r + b_1 y_r + c_1 z_r = d_1 \qquad a_2 x_r + b_2 y_r + c_2 z_r = d_2\,, \quad (5.14)$$

où :

$$\begin{aligned} a_1 &= \alpha_u r_{11} - r_{13}(u - u_0) & a_2 &= \alpha_v r_{12} - r_{13}(v - v_0) \\ b_1 &= \alpha_u r_{21} - r_{23}(u - u_0) & b_2 &= \alpha_v r_{22} - r_{23}(v - v_0) \\ c_1 &= \alpha_u r_{31} - r_{33}(u - u_0) & c_2 &= \alpha_v r_{32} - r_{33}(v - v_0) \\ d_1 &= t'_3(u - u0) - \alpha_u t'_1 & d_2 &= t'_3(v - v0) - \alpha_v t'_2\,, \end{aligned} \quad (5.15)$$

où t'_1, t'_2 et t'_3 sont les éléments de la dernière colonne de T^{-1}. Ayant deux équations avec trois inconnues, on ajoute la troisième équation pour pouvoir résoudre le système : l´hypothèse de sol plat $z = 0$. Ainsi les coordonnées recherchées sont :

$$x_r = \frac{\left(d_1 - \left(\frac{b_1}{b_2}\right) d_2\right) - \left(c_1 - \left(\frac{b_1}{b_2}\right) c_2\right) z}{a_1 - \left(\frac{b_1}{b_2}\right) a_2} \qquad y_r = \frac{\left(d_1 - \left(\frac{a_1}{a_2}\right) d_2\right) - \left(c_1 - \left(\frac{a_1}{a_2}\right) c_2\right) z}{b_1 - \left(\frac{a_1}{a_2}\right) b_2} \quad (5.16)$$

La liste des x_r et y_r constitue maintenant la trajectoire projetée sur le sol.

5.4 Planification des mouvements

Le premier auteur à établir une solution au problème du *suivi d'une trajectoire* a été Dubins [Dubins 57] qui a démontré que la trajectoire optimale (en longueur) entre deux points consistait toujours en une séquence de lignes droites et arcs de cercle. Reeds a ajouté les déplacements en arrière [Reeds 90] à ce modèle. Dans ces résultats chaque élément de la trajectoires est C^2 mais la courbure est discontinue entre deux des morceaux de la trajectoire, ainsi un robot réel devra s'arrêter à chaque intersection pour assurer la continuité des vitesses linéaire et angulaire. Pour surmonter cette difficulté, l'approximation par des clothoïdes des séquences ligne droite-arc de cercle a été proposée mais le contrôle du mouvement du robot sur une telle courbe s'avère difficile.

Une autre approche [Lamiraux 01] suggère la modélisation du robot en 4 dimensions au lieu de trois (*cf.* équations 5.17) associant à chaque configuration un angle de direction associé à la courbure ($\kappa = tan\,\zeta$). Une fonction croissante $\alpha(t)$ permet le passage progressif d'une configuration à l'autre (*cf.* équation 5.18). Néanmoins cette conception a besoin d'une étape de planification préliminaire pour établir les configurations à relier. Notons que toutes ces solutions sont sous-optimales, le calcul des trajectoires de longueur minimale est encore aujourd'hui un problème ouvert.

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}cos\theta\\ sin\theta\\ 0\end{pmatrix}\tilde{v} + \begin{pmatrix}0\\ 0\\ 1\end{pmatrix}\tilde{w}\,, \qquad \begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\zeta}\end{pmatrix} = \begin{pmatrix}cos\zeta\, cos\theta\\ cos\zeta\, sin\theta\\ sin\zeta\\ 0\end{pmatrix} v + \begin{pmatrix}0\\ 0\\ 0\\ 1\end{pmatrix} w \tag{5.17}$$

$$P(t) = (1-\alpha(t))\gamma(X_1,t) + \alpha(t)(X_2, t-1) \tag{5.18}$$

Dans notre cas, nous préservons le modèle du robot en 3D (équation gauche 5.17). Nous avons ainsi choisi l'utilisation des arcs de cercle et des lignes droites qui relient la séquence de points obtenus après la projection sur le sol pour obtenir une trajectoire effective. Notons que s'il y a des occultations (*e.g.*, les ombres), la trajectoire suivi par le robot ne sera pas au milieu du chemin

5.4.1 Déplacement par arcs de cercle

Commençons par décrire la géométrie de l´arc de cercle (voir figure 5.10). Les points de départ et d´arrivée (dans le repère robot) peuvent être reliés par deux arcs de cercle. L'angle α et le rayon R qui les déterminent sont obtenus en traçant deux lignes : l du centre du robot au point destination, et m, perpendiculaire a l, connectant son point milieu et l'axe Y.

L'équation qui décrit m est donnée par :

$$y = -\left(\frac{x_1 - x_0}{y_1 - y_0}\right)\left(x - \frac{x_0 + x_1}{2}\right) + \frac{y_0 + y_1}{2} \tag{5.19}$$

d'où le centre de l'arc de cercle C :

$$C(x_c, y_c) = \left(\frac{x_1 - x_0}{y_1 - y_0}\right)\frac{x_1 - x_0}{2} + \frac{y_0 + y_1}{2} \tag{5.20}$$

Le rayon R de l'arc et l'angle α sont alors représentés par $R = \|y_c - y_0\|$ et :

$$\alpha = \begin{cases} \arcsin\left(\frac{x_1 - x_0}{2R}\right) & \text{si } |y_1 - y_0| < |x_1 - x_0| \\ \arcsin\left(\frac{x_1 - x_0}{2R}\right) + 2\pi & \text{si } |y_1 - y_0| > |x_1 - x_0| \text{ et } x_0 < 0 \\ \arcsin\left(\frac{x_1 - x_0}{2R}\right) - \pi & \text{dans les autres cas}\,. \end{cases} \tag{5.21}$$

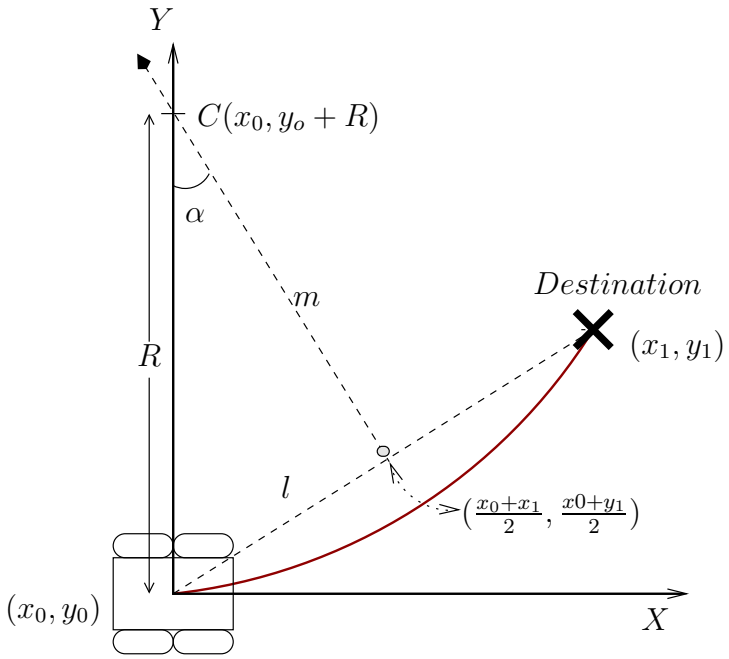


FIG. 5.10 – Arc de cercle

La condition sur α garantit le choix de l'arc qui va vers l'avant. Par définition les vitesses linéaire et angulaire sont respectivement $v = w\,R$ et $w = \frac{\alpha}{t}$. Ce système d'équations concernant (v, w et t) est résolu en fixant v, ce qui permet en même temps d'avoir un mouvement uniforme.

Les consignes ainsi obtenues ne peuvent pas toujours être exécutées par le robot. Selon la plate-forme il existe une courbure maximale qu'il sera capable de suivre. Dans notre cas, un seuil contraint l´angle α : au-delà de cette limite la configuration est considérée non-valide et ignorée ; si c'est le cas, le robot va s'asservir sur le point suivant de la trajectoire.

La connexion des points par des arcs de cercle a une tendance à former des trajectoires d'une courbure élevé, surtout quand les points sont très proches. Une trajectoire plus naturelle est obtenue en représentant les points originaux de la trajectoire par une courbe de Bézier cubique (cette méthode est applicable aussi au lissage de contours *cf.* § 4.3.3) [Pettre 03]. Cette approximation rapide est valable car le robot n'a pas besoin de passer exactement par les points projetés.

5.4.2 Fusion temporelle des trajectoires

La méthode décrite jusque là fonctionne très bien pour des séquences d'images prises dans les mêmes conditions d'illumination et quand les régions sont assez distinctes pour que la segmentation les sépare correctement. Quand ce n´est pas le cas, le contour peut varier d'une image à l'autre d'une manière abrupte ; si nous remplaçons systématiquement les trajectoires à chaque itération, nous pourrions substituer une trajectoire bien planifiée avec une autre fausse. Dans le but de prévenir ces situations, on peut tenir compte des derniers résultats pour prédire la nouvelle trajectoire et fusionner ces estimations à la trajectoire courante extraite directement du contour.

Le principe d'estimation considère que les points qui n´ont pas encore été parcourus par le robot deviennent des prédictions qui devraient être confirmées par les nouveaux arrivants. Une moyenne pondérée des points prédits et courants est faite pour un certain nombre de regions

circulaires concentriques autour du repère principal du robot. Le résultat est une séquence de trajectoires plus continue [Mateus 05].

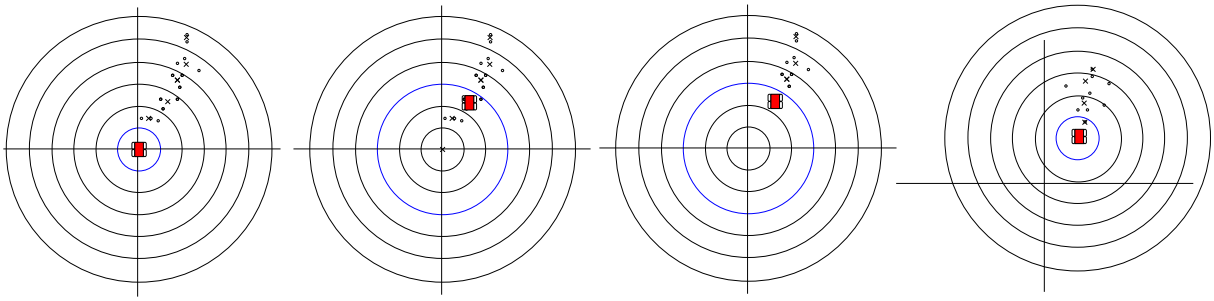


FIG. 5.11 – Fusion des points résultants des différentes acquisitions sur une seule trajectoire.

La liste des points trajectoire (en coordonnées globales) calculée lors les n[33] derniers traitements est stockée. Quand une nouvelle image est acquise et la trajectoire correspondante calculée, la liste est mise à jour. Les points sont oubliés après n itérations ou bien quand le robot passe près d'eux (distance entre le robot et le point est inférieure à 1 m). Pour fusionner les points restants, des anneaux concentriques sont « dessinés » autour du robot, définissant le rayon d'action de chaque point. Tous les points sur le même anneau sont fusionnés ; chacun participe avec un facteur de pondération qui dépend de l'écart à la moyenne de tous les points dans l'anneau.

Cette procédure de fusion est illustrée en figure 5.11 ; à gauche, la situation en fin de traitement dans une itération ; les cercles sont centrées dans la position courante du robot ; à l'état suivant, le système a effectué une partie de la trajectoire et se prépare pour la fusion ; il supprime les points de trajectoire qu'il a dépassés et centre les cercles sur sa position courante.

5.5 Suivi d'objets par le modèle de contour actifs

Le modèle des contours actifs est exploité afin de percevoir l'évolution dynamique des objets dans la scène. Il s'agit de détecter et de suivre, en temps réel le chemin ou d'autres objets vus par le système de vision du robot.

5.5.1 Contours actifs

Nous avons utilisé la technique de suivi de contours développée par Kass [Kass 88], connue sous l'appelation par *contours actifs* ou *snakes*. Notre implémentation est basée sur une méthode particulière de contours actifs, appelée « Electric Snakes », introduite par Antonio Marin [Marin-Hernandez 04] dans sa thèse de doctorat.

Cette méthode décrit le contour d'un objet par un modèle élastique déformable fermé, passant par une suite de points de contrôle $\boldsymbol{v}$ soumis simultanément à un ensemble de forces issues de potentiels d'énergies internes et externes. Ce modèle évolue en minimisant ces énergies jusqu'à l'obtention d'un état d'équilibre, qui représente le meilleur ajustement du modèle à l'environnement en terme de position, d'orientation et de déformation du contour. Pour cela, les forces sont définies en fonction de la forme souhaitée pour le contour de l'objet suivi dans l'image [Marin-Hernandez 99].

[33] Paramètre à régler

Généralement, la minimisation d'énergie est restreinte aux lignes perpendiculaires à la courbe sur chaque point de contrôle, mais cela limite les possibilités de déformation du contour. Dans le cas où l'énergie est évalué en tout voisin de chaque point de contrôle, après quelques itérations, ces points pourraient « glisser » le long du contour et se retrouver groupés aléatoirement en quelques endroits le long de la courbe ; le contour final pourrait être alors très différent de la forme désirée. La méthode proposée par A.Marin permet de limiter ce processus en introduisant une autre force interne sur les points de contrôle : pour définir cette force, le contour est traité comme un conducteur électrique ayant une charge électrique constante Q. Cette charge produit une nouvelle force de répulsion entre points de contrôle, qui les redistribue tout le long de la courbe, avec une densité proportionnelle à la courbure [Avina-Cervantes 03a].

5.5.1.1 Formulation des énergies

Une courbe continue est obtenue en utilisant une interpolation des points de contrôle (définissant le contour actif $\boldsymbol{v}$) par des fonctions B-splines (quadratiques ou cubiques). L´énergie totale E_{tot} pour un contour actif paramétrique $\boldsymbol{v} = (x(s), y(s))$ est donnée par :

$$E_{tot} = \int E_{int}\left(\boldsymbol{v}(s)\right) + E_{ext}\left(\boldsymbol{v}(s)\right)\, ds\,, \tag{5.22}$$

où s représente l'abscisse curviligne, E_{int} et E_{ext} regroupent respectivement les énergies internes et externes. Classiquement, l'énergie interne est définie par l'équation :

$$E_{int}(\boldsymbol{v}) = \int_0^1 \omega_1(s)\boldsymbol{v}_s^2 + \omega_1(s)\boldsymbol{v}_{ss}^2\, ds \tag{5.23}$$

et l'énergie externe est donnée par :

$$E_{ext}(\boldsymbol{v}) = \int_0^1 P\left(\boldsymbol{v}_s\right)\, ds \tag{5.24}$$

où les indices inférieurs sur $\boldsymbol{v}$ dénotent l'ordre de dérivation, $w_1(s)$ et $w_2(s)$ sont des facteurs de poids assignés aux termes d'élasticité et de lissage respectivement, et P est l'intensité du potentiel externe. Généralement, ce potentiel P est proportionnel au gradient de l'intensité lumineuse dans l'image, de sorte que le contour actif se colle sur les discontinuités de luminance.

La nouvelle force de répulsion entre points de contrôle [Marin-Hernandez 99], qui permet d'obtenir une meilleure stabilité, rajoute une nouveau terme dans l'énergie interne alors définie comme :

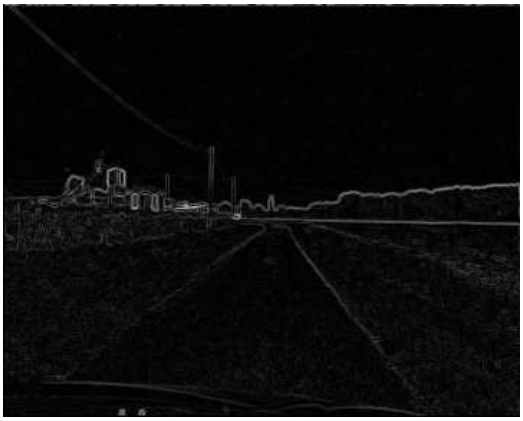
$$E_{int}(\boldsymbol{v}) = \int_0^1 \omega_1(s)\boldsymbol{v}_s^2 + \omega_1(s)\boldsymbol{v}_{ss}^2 + k\frac{\Sigma^2}{\boldsymbol{v}_s^2}\, ds \tag{5.25}$$

où Σ est la densité de charge électrique, et k un coefficient constant.

Le terme $k\Sigma^2$ peut être vu comme une constante, une fois que la charge électrique a été assignée. En plus, une fois l'équilibre atteint, la distribution de la densité de charge ne change pas pour un conducteur électrique isolé, quelle que soit sa position dans l'espace.

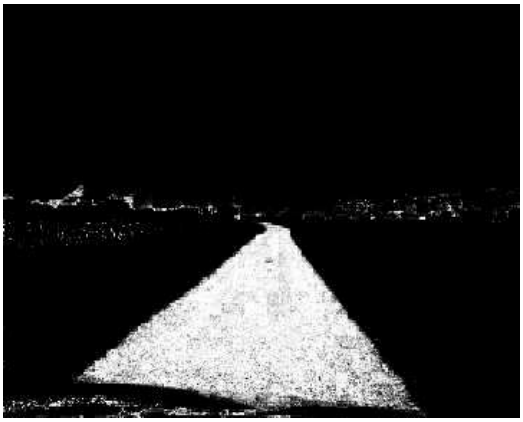
5.5.1.2 Potentiel externe pour des *Snakes* couleur

La force externe qui attire le *Snake* vers le contour que l'on souhaite extraire, est couramment définie comme une fonction du gradient calculé sur l'image d'intensité. Dans des images d'environnements complexes, ni les gradients sur l'intensité, ni les gradients sur les composantes

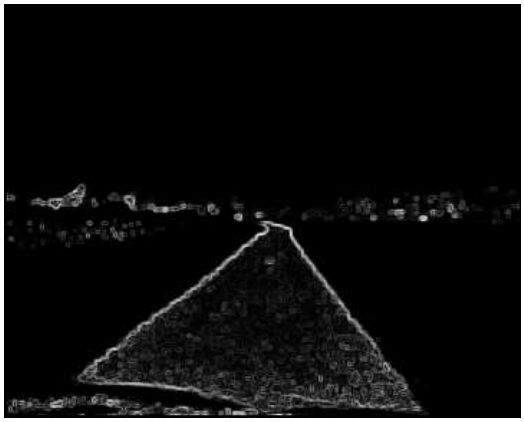


(a) Le gradient sur l'intensité

(b) Le gradient sur la couleur



(c) L'espace de couleur RGB normalisé



(d) Le gradient sur la couleur focalisé

FIG. 5.12 – Les champs de potentiel externes utilisés dans les contours actifs couleur

couleur sont suffisamment robustes pour garantir une bonne stabilité et une convergence rapide, qui attire le contour actif sur les contours de la cible (*cf.* figure 5.12 a), b)).

Dans [Marin-Hernandez 04], nous présentons une étude comparative de plusieurs gradients couleur utilisés dans le domaine du suivi des objets colorés. Finalement, nous avons proposé un gradient couleur focalisé, très efficace pour le suivi d'objets appris au préalable. Ce gradient est focalisé, car il est appliqué sur une image calculée par un filtrage gaussien dans l'espace RGB normalisé [Avina-Cervantes 03a], en exploitant les statistiques obtenues dans la phase de caractérisation des régions ; nous utilisons les caractéristiques apprises préalablement à partir des échantillons de la base de données, pour les objets de la même classe que celui qui est suivi (chemin, arbre, ...), plus spécifiquement leur couleur moyenne et leur écart type.

Les figures 5.12 c) et d) montrent les excellents résultats obtenus par l'utilisation de ce gradient couleur focalisé sur la classe chemin. Même si la région chemin n'est pas complètement isolée du reste des régions, les résultats sont nettement meilleurs qu'avec d'autres définitions de gradient. Il est possible de trouver (par notre méthode de modélisation de la scène) les bons paramètres nécessaires pour effectuer le suivi d'une cible dans un espace de caractéristiques bien défini.

5.5.1.3 Complexité algorithmique

Étant donné que nous avons appliqué la programmation dynamique pour résoudre le problème de la minimisation d'énergie par l'équation de mouvement [Amini 90], le nombre des points de contrôle qu'on prend en compte à l'initialisation du suivi, doit être un compromis entre un nombre réduit (pour limiter la complexité de la méthode) et un nombre suffisant (pour obtenir une bonne description du contour de la cible).

A chaque itération, la complexité algorithmique de la méthode de programmation dynamique est $O(nm^{k+1})$ où n est le nombre de points de contrôle, m est le nombre de positions possibles pour un point (typiquement, 9 positions), et k est l'ordre de dérivation le plus élevée pour le contour (typiquement, 2). Nous accélérons cette étape en utilisant une approche dans un espace multi-échelle, ce qui permet par ailleurs de rendre la méthode plus robuste aux minima locaux que les approches variationnelles.

5.5.2 Suivi des objets par intégration de deux processus visuels

Notre approche exploite une collaboration entre deux fonctions visuelles, traitant respectivement de la reconnaissance des objets présents sur une image initiale et du suivi de certains d'entre eux sur une séquence postérieure.

Dans une première étape, le contour actif est défini par une position initiale de la courbe $\boldsymbol{v}$. C'est peut-être la phase la plus importante sur laquelle repose le suivi des objets. Dans notre application au suivi d'un chemin, cette initialisation est faite automatiquement avec une région délimitée par un nombre fixe de points de contrôle ($n = 50$), définissant ainsi une zone de recherche. Pour cela, nous profitons des résultats issus de l'étape d'extraction de chemin (échantillonnage et lissage de la frontière de l'objet) décrite précédemment.

A partir de cette position initiale, le contour va évoluer, soit dans une même image (segmentation d'un objet), soit dans des images successives d'une séquence (suivi d'un objet) : les forces sont appliquées sur les points de contrôle, qui à chaque itération peuvent se déplacer dans un voisinage donné (typiquement, dans une fenêtre 3x3). L'évolution du contour actif est déterminée par son équation de mouvement, dérivée à partir de l'équation 5.22.

Dans le cas du suivi d'une route ou d'un chemin, une réinitialisation du *Snake* décrivant le contour à suivre, est nécessaire périodiquement car même avec le gradient focalisé, le contour peut être attiré par certains objets étrangers à la route. Par exemple, des objets au bord de la route ou des forts ombrages dus à des poteaux ou des arbres. Également, les *Snakes* peuvent être déformés d'une manière inappropriée, rendant très difficile la récupération du contour désiré.

5.5.3 Architecture de coopération visuelle

Ce sont des expériences sur des séquences d'images réelles qui ont permis de valider la méthode *Snake* couleur développée et de mettre en relief en même temps, l'importance d'une coopération adéquate (problème de la synchronisation) entre les systèmes visuels impliqués. Dans la figure 5.13, nous présentons la modélisation de la scène, l'extraction et le suivi d'une route depuis un véhicule se déplaçant en rase campagne. Bien que les images analysées montrent une visible dégradation de la texture du fait de la vibration et de la vitesse du véhicule, l'extraction du chemin est acceptable. Dans nos expériences avec la séquence vidéo, une image couleur de 360×288 pixels a été analysée par l'algorithme de segmentation de couleur dans environ 100 ms sur une station Sparc 5. Les images segmentées après l'obtention du modèle global 2D sont montrées du côté gauche sur la figure 5.13. Du côté droit, nous présentons les évolutions du suivi. Dans cette séquence d'images, nous notons que seulement quelques images du côté gauche

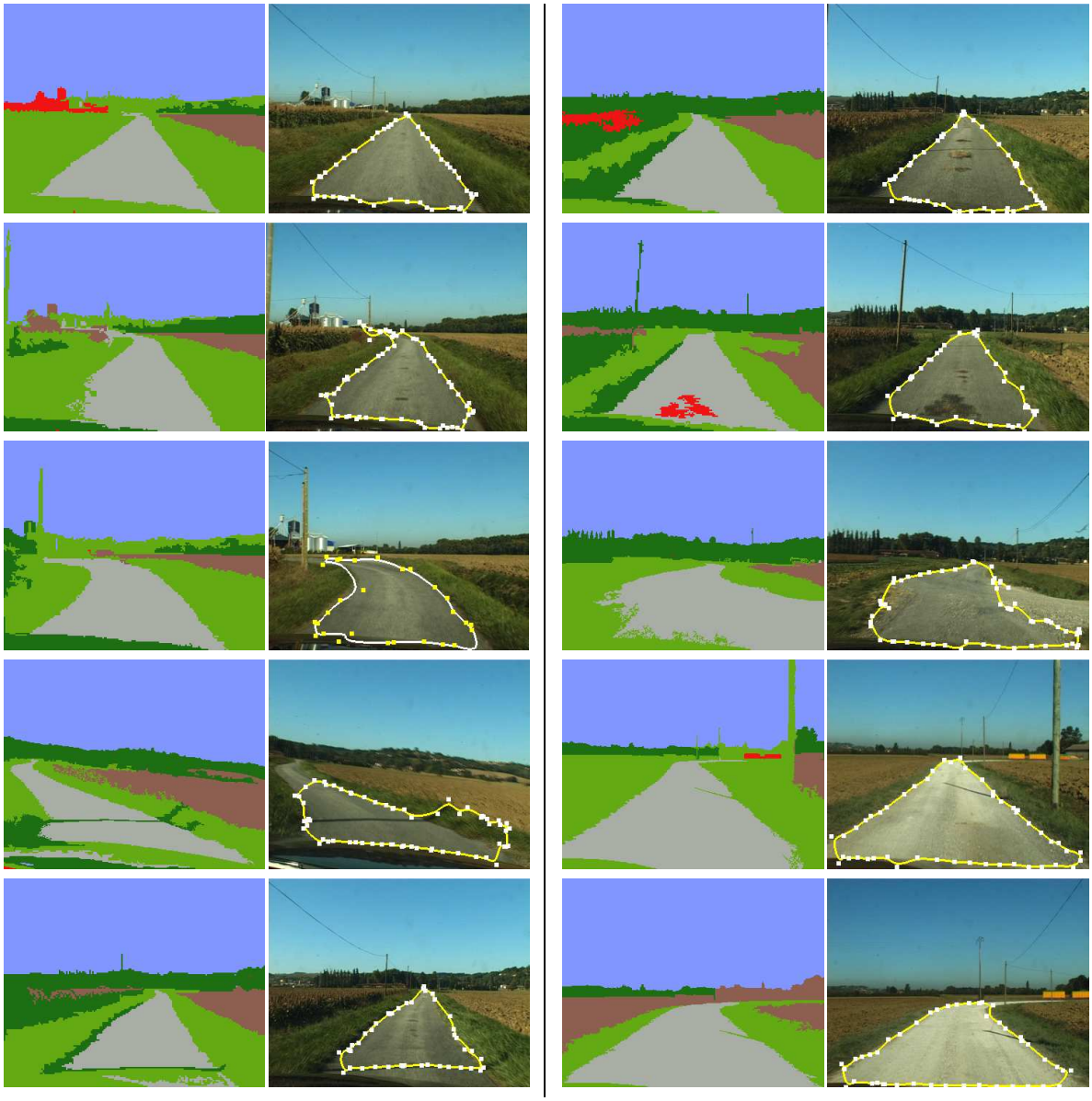


FIG. 5.13 – Extraction et suivi des chemins dans une séquence vidéo

sont employées comme initialisation du résultat présenté à droite. Les zones rouges sur les images à gauche représentent des *outliers* ou des régions « non classifiés ».

L'architecture de coopération entre les modules du suivi et d'extraction de chemins est montrée dans la figure 5.14.

L'extraction de la cible (module noté R-LOC) est effectuée en continue, mais du fait de sa complexité algorithmique, elle fonctionne à une fréquence relativement faible (autour de 1 Hz) ; la procédure de suivi (module noté R-TRACK) est prioritaire, et de ce fait, a une fréquence 10 fois supérieure ($10 \sim 16$ Hz). Notons que, si une boucle de contrôle n'exploitait que la fonction d'extraction du chemin (R-LOC) dans un processus de navigation, une telle période de temps serait insuffisante : la coopération avec la fonction de *Tracking* du chemin (R-TRACK) permet de surmonter cet inconvénient (*cf.* figure 5.14). Il sera de plus possible toutes les 10 itérations du suivi, de corriger une dérive du *Tracking*, ou d'adapter les caractéristiques de la cible et de

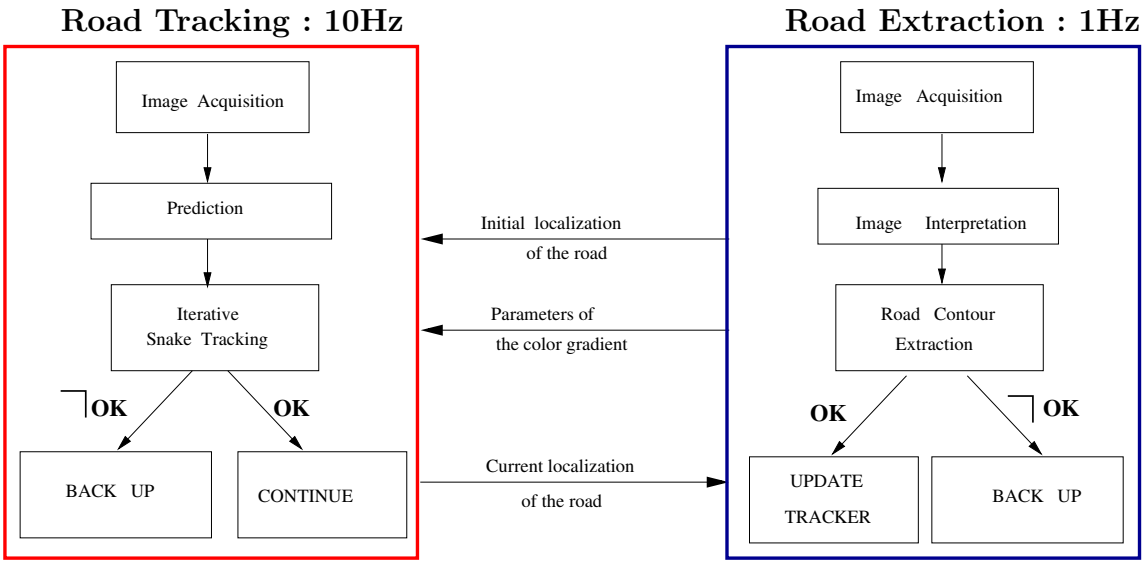


FIG. 5.14 – Architecture de coopération entre le module d'extraction de chemins et celui du suivi d'objets

prendre ainsi en compte des variations sur les conditions d'illumination.

Dans le groupe RIA, nous avons déjà analysé une telle coopération pour la navigation des robots d'intérieur en utilisant seulement les localisations relatives [Ayala-Ramirez 00] ou pour les robots extérieurs, pendant une tâche de modélisation [Avina-Cervantes 02].

Récapitulons les différents avantages d'une telle coopération, supposant que R-TRACK peut traiter 10 images quand R-LOC peut seulement en analyser une :

- ▷ Au début, sur l'image I_0, R-LOC trouve la position de la route. Cette position est envoyée comme le contour initial de la route à R-TRACK ; les paramètres de couleur sont également transmis, de telle sorte que R-TRACK les exploite pour le filtrage gaussien préalable au calcul du gradient couleur focalisé.
- ▷ chaque 1 s à partir de l'image I_{10}, puis pour une image sur dix, R-LOC donne une nouvelle localisation de la route, en utilisant l'interprétation globale de la scène. Cette position globale permet (1) de détecter les erreurs du *tracker* et de réinitialiser R-TRACK, et (2) de mettre à jour les paramètres du gradient.

La principal limitation des *Snakes* est l'initialisation. D'une part, elle est généralement fournie par un professeur (typiquement, un opérateur de manière intéractive), de l'autre elle doit être faite le plus près possible de la solution. Dans notre implémentation, le module d'extraction du chemin (ou d'autres objets à suivre) à partir de la description 2D de la scène, fournit une initialisation automatique des contours actifs.

Les *Snakes* présentent d'autres limitations qui sont associées aux problèmes d'occultation partielle d'objets, ce qui est mal géré par cette technique. Ainsi, des difficultés de convergence apparaissent lorsque l'objet suivi est constitué par des régions concaves. Pour pallier ces inconvénients nous exploitons la réinitialisation périodique du processus de suivi par le module d'extraction de chemins. Cela donne une grand stabilité à notre système qui est gênée uniquement par des problèmes de synchronisation temporelle entre ces deux derniers modules.

Dans notre implémentation, la réinitialisation du *Tracking* est donc faite périodiquement ; quelquesoit l'état du suivi, le contour actif est réinitialisé toutes les 10 itérations environ. Il serait meilleur de disposer d'un critère probabiliste qui évalue le bon positionnement du contour

actif sur les bords de la cible. Ce test pourrait exploiter une comparaison entre les moments d'ordre supérieur calculés pour la surface délimitée par le contour actif, et les moments appris au préalable ; il pourrait prendre en compte les dernières itérations afin de détecter rigoureusement les changements trop abrupts dans la forme du contour.

5.6 Expérimentations

Les algorithmes décrits dans les sections précédentes ont été développés et validés sur le robot ATRV appelé Dala. Il dispose d'une platine stéréo mobile avec deux caméras Sony Micropix C-1024, d'une caméra panoramique, d'un télémètre laser et de 14 capteurs ultrasons. Dans nos travaux seule une caméra est exploitée.

Nos modules fonctionnels sont intégrés dans l'architecture LAAS [Alami 98] et pour ce faire, nous exploitons le logiciel GenOM[34]. Constituée de trois couches, cette architecture prend en compte toutes les composantes nécessaires sur un robot, pour effectuer la planification, l'exécution et la supervision des tâches et pour obtenir finalement, un système robuste adapté à l'autonomie et au fonctionnement temps réel du robot (*cf.* figure 5.14).

5.6.1 Intégration

Les contraintes d´exécution associées à l´autonomie du robot et à la navigation en temps réel impliquent une utilisation judicieuse des ressources du robot et la prise en compte des possibles failles de chaque composant du système ou de la communication entre eux. L'intéraction entre les diverses fonctionnalités inhérentes au déplacement du robot sont décrites ci-après.

Dès le début, l'acquisition des images s'exécute en mode périodique. La dernière image acquise est sauvegardée dans un poster (espace local de mémoire, partagé entre plusieurs modules clients) et mise à disposition des autres modules. En fonction de la mission de plus haut niveau que le robot doit effectuer (par exemple, *Va de la ferme au champ*), le superviseur décide la prochaine tâche à être exécutée (par exemple, *Détecter le chemin*, puis *Suivre un chemin*) ; la primitive de mouvement pertinente est alors déclenchée. Le module *Road* (en charge d'extraire les régions propres à la navigation) lit l'image et effectue la procédure de segmentation et classification des régions ; selon la primitive à exécuter, les contours de certaines régions seront extraits et exportés dans un autre poster avec quelques informations additionnelles concernant l'image traitée. Dès que la procédure sur cette image est finie, *Road* lit à nouveau le poster exporté par le module *Caméra* et le traitement se répète jusqu'à la fin de la primitive.

Le module *SBM* (Sensor Based Motion) attend les résultats du traitement visuel ; lorsqu´ils deviennent disponibles dans le poster produit par *Road*, les contours, exprimés comme les points de contrôle d'une B-spline fermée, sont reconstruits. Ensuite, le module extrait une trajectoire provisoire dans l'image, trajectoire que le robot devra suivre jusqu'à ce que le poster qui contient les contours soit mis à jour soit par la segmentation soit par le tracking visuel.

La série de points dans l´image qui forment la trajectoire est ensuite transformée en une trajectoire 2D sur le plan du sol, exprimée dans le repère du robot (au milieu des quatre roues) en faisant la supposition du sol plat, tel que ce fut décrit dans la section 5.3.4. Cette transformation exploite des données produites par le module *Insitu* sur les transformations entre repères internes du robot (transformation affine image-caméra ; transformations homogènes caméra-platine, platine-robot...). Finalement, tous les points des trajectoires calculés depuis les dernières

[34]GenOM, Generator of Modules, un logiciel développé au sein du groupe RIA afin de réaliser la couche fonctionnelle des robots

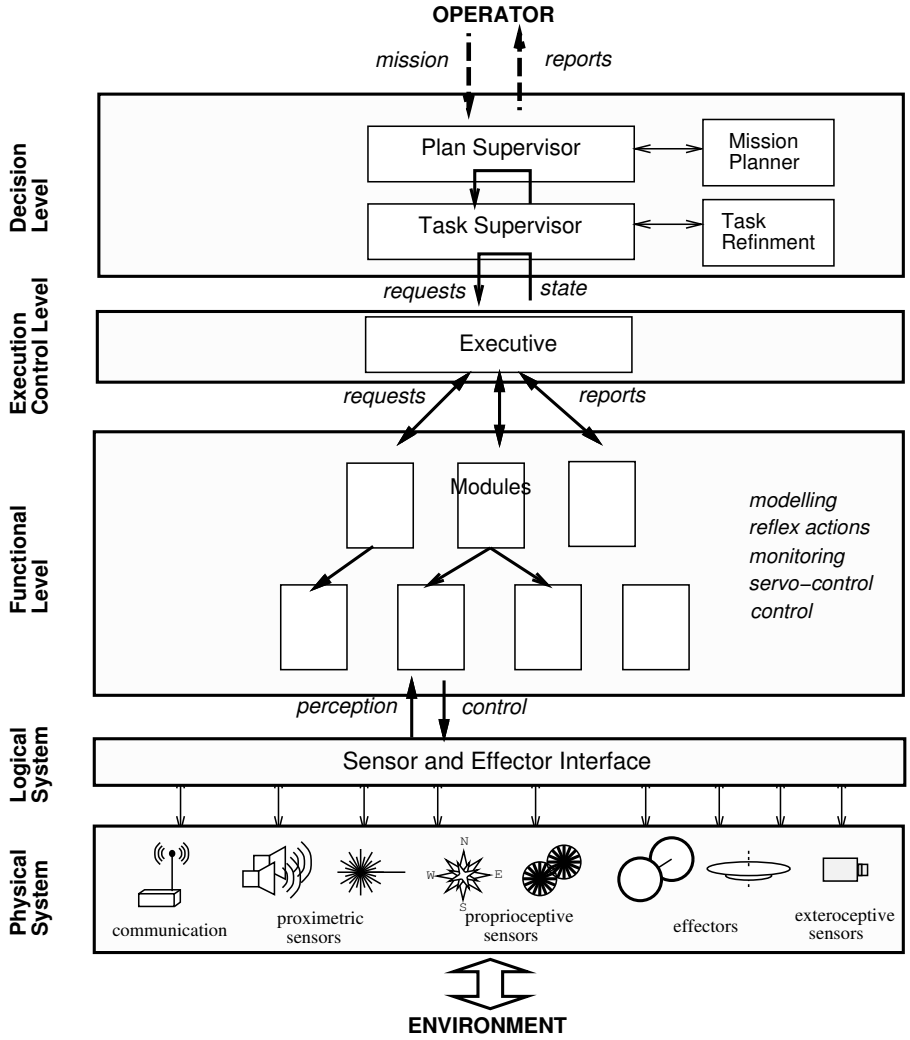


FIG. 5.15 – Architecture LAAS

images, sont mis à jour, pour être exprimés dans le repère courant du robot, grâce aux *déplacements relatifs du robot* depuis sa position courante et ses positions lors des dernières acquisitions. Ce déplacement, estimé par odométrie, est obtenu en lisant le poster produit par le module *PoM*.

A partir de ces points exprimées dans le repère courant du robot, la méthode de fusion temporelle décrite en section 5.4.2 est appliquée pour générer une trajectoire unique. Un arc de cercle reliant la position courante du robot avec le premier point de la trajectoire, est calculé et les vitesses linéaire et angulaire en sont déduites. Ces consignes sont exportées par le module *SBM*. Le module de commande de moteurs *Rflex* en mode de suivi de vitesses lit et exécute périodiquement les consignes de déplacement.

Dans certaines des primitives où la platine doit bouger pour préserver la cible dans le champ de vue, le principe pour fournir les consignes est le même. Le module *SBM* calcule aussi les consignes de la platine et met à jour un poster avec cette information ; le module *Platine* en mode suivi de position ou vitesse, lit périodiquement le poster et effectue les déplacements de la caméra.

Le calcul d'une trajectoire provisoire depuis chaque image acquise est censé maintenir une

consigne cohérente pendant le temps de calcul du traitement visuel. Toutefois, la trajectoire est entachée des erreurs provenant de l'odométrie et de la supposition du sol plat. Le seul moyen de corriger ces erreurs, est de traiter une nouvelle image acquise depuis la position courante ; plus la segmentation est rapide, plus la trajectoire sera précise. Évidemment, on peut activer en parallèle aussi la fonction de suivi visuel par contours actifs pour augmenter la fréquence à laquelle les données de la boucle hiérarchique de commande sont mises à jour.

5.6.2 Résultats expérimentaux

En général les primitives de déplacement « Suivre un chemin », « Suivre une bordure », et « Aller vers objet » fonctionnent assez bien. Cependant, nous avons surtout validé le suivi de chemin, et réalisé une quantité limitée de tests de validation des autres primitives, principalement à cause de la difficulté d'effectuer des expérimentations sur un terrain adéquat (milieu agricole). Dans ces expériences, le robot se déplace naturellement en continu tel que prévu car l'utilisation d'un lissage par courbes de Bézier réduit la longueur des trajectoires et améliore la continuité de la commande. La figure 5.16 montre une simulation qui illustre les avantages inhérents à l'utilisation de ces courbes pour la planification de la trajectoire à suivre.

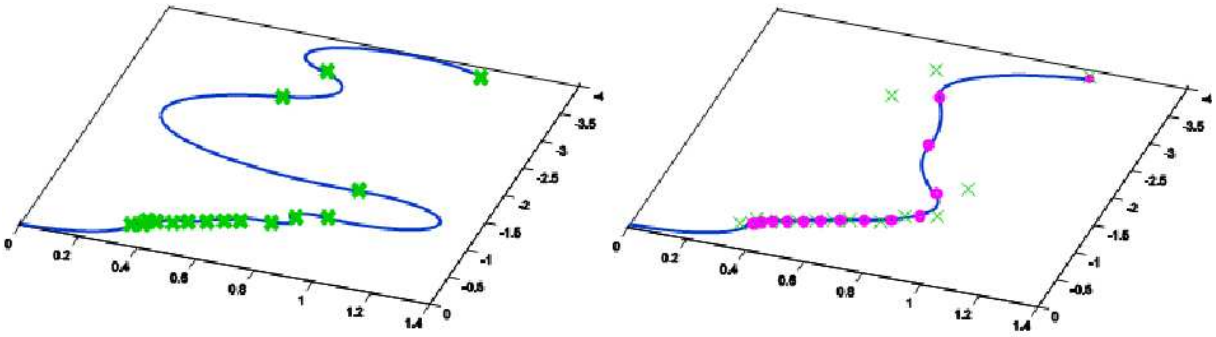


FIG. 5.16 – Trajectoires, avant et après utilisation d'une approximation par les courbes de Bézier

La figure 5.17 illustre deux primitives de navigation, à gauche « Suivre un chemin » et a droite « Suivre une bordure ». Nous présentons en même temps deux perceptions du mouvement, celle du robot (*cf.* images 5.17.a et 5.17.c) et celle d'un observateur extérieur (*cf.* images 5.17.b et 5.17.d). Dans ces images sont dessinés les points de contour issus du module d'extraction de chemins *Road* et au centre (ou à côté d'une bordure) les points estimés non lissés pour la trajectoire à exécuter.

La fusion temporelle des trajectoires s'est révélée extrêmement utile. Elle garantit presque toujours qu'il existe une trajectoire à suivre et elle évite la prise en compte (ou diminue les effets) des trajectoires issues d'une segmentation erronée. En même temps, la fusion réduit les changements de courbure entre deux trajectoires planifiées et le mouvement résultant est plus naturel.

Nous avons détecté quelques inconvénients qui ont posé la plupart des difficultés. La primitive « Suivre un chemin » se réalise sans complications tant que les deux bordures (réelles) de la voie sont présentes mais cela n´est pas nécessairement le cas. Ainsi, le chemin « visualisé » par le robot peut être incomplet (champ de vue limité), par exemple la situation où les limites de l'image sont considérées comme les contours d'un chemin qui est de plus en plus large (*cf.* 5.18(a)). Tant qu´il reste des segments lointains où les croisements entre lignes horizontales et contour de la région chemin dans l'image, sont valables, le robot aura toujours une référence à

FIG. 5.17 – Résultats pour deux primitives de déplacement. a) et c) les contours non lissés aperçus par le robot, b) et d) le point de vue d'un observateur

suivre. Dans cette situation, le robot navigue au milieu d'un chemin imposé partiellement par les bords de l'image acquise jusqu'à se rapprocher d'une trajectoire imposée vraiment par la forme du chemin. La figure 5.18(b) illustre l'utilisation du bord droit de l'image pour guider le robot. Notons que pour corriger cela un asservissement visuel d'une caméra active plaçant le chemin au centre de l'image serait essentiel ; une caméra à focale variable pourrait être contrôlée pour adapter son champ de vue afin d'avoir toujours les bords du chemin dans l'image.

La figure 5.19 illustre le suivi correct d'un chemin goudronné (parking du LAAS) même si le système de reconnaissance de chemins a été conçu pour détecter des chemins de terre, *i.e.*, le système de classification ne connaît pas correctement la classe *goudron*. De plus, il est évident qu'un module de vision monoculaire basé sur l'information couleur et texture n'est pas approprié pour naviguer dans un environnement du type parking. Néanmoins cette expérience nous a permis de valider d'avantage notre méthode. La séquence commence d'abord avec un chemin assez large en se guidant principalement par les points lointains. Notons que certains contours peuvent être apparemment décalés par rapport à l'image actuelle ; ceci est provoqué par le décalage entre l'image depuis laquelle le chemin est extrait, et l'image sur laquelle le contour extrait est affiché ; ce décalage vient de la transmission de l'image, plus lente que la transmission

(a) Bordes proches du chemin non détectés (b) Bord droit du chemin non détecté

FIG. 5.18 – Problèmes de navigation liés au champ de vue limité du robot

d'une liste de points au système d'affichage.

Dès que le chemin commence à se centrer sur l'image, la présence d'ombres provoque une modification des contours effectifs du chemin ; ce problème est directement lié aux caractéristiques de notre caméra dont le gain automatique fait des dégâts sur le contraste des images (des études à ce sujet seront nécessaires). En outre, observons que le robot n'est pas conduit exactement au milieu du chemin *réel*, ce qui s'explique par son champ de vue limité. Dans la même séquence, nous montrons les inconvénients de se placer en face de la source lumineuse (ici le soleil), car cela provoque la saturation du capteur dans certaines portions d'images (le robot peut aussi être ébloui), ce qui gêne en même temps la fonction de navigation. Enfin, la trajectoire s'est conclue d'une manière satisfaisante dans l'actuel chantier du LAAS puisque le robot a parcouru (à très faible vitesse) plus de 100 m en complète autonomie.

5.6.3 Défaillances du module visuel

Dans des milieux extérieurs, les changements continus d´illumination peuvent poser de nombreux problèmes aux techniques de segmentation et reconnaissance visuelles. Sous certaines conditions, la couleur apparente des objets est modifiée, de sorte que les vecteurs d´information colorimétrique ne sont plus semblables à ceux qui font partie de la base d´apprentissage et qui correspondent au même objet sous d´autres conditions de luminosité.

La méthode de calibration en couleur automatique appliquée après la reconstruction de l´image et la prise en compte des mesures sur la texture sont censées corriger ces variations. Néanmoins, il existe toujours des sources d´erreur ; par exemple, il n'est pas possible d´empêcher que des objets voisins proches dans l´espace couleur soient fusionnées, surtout quand l´un d´entre eux est loin du robot (l´information de texture est pauvre dans ces cas). La figure 5.20(a) illustre parfaitement ce dernier effet ; le mur du bâtiment est agrégé à la route. Une autre source typique d´erreurs est la présence d'ombres : les deux cas présentés en figure 5.20(b) et 5.20(c), seront facilement détectables et corrigés en complétant l'analyse à l'aide des capteurs qui fournissent des informations de profondeur (vision stéréo).

De plus, le temps de calcul de la procédure de segmentation et reconnaissance est autour de

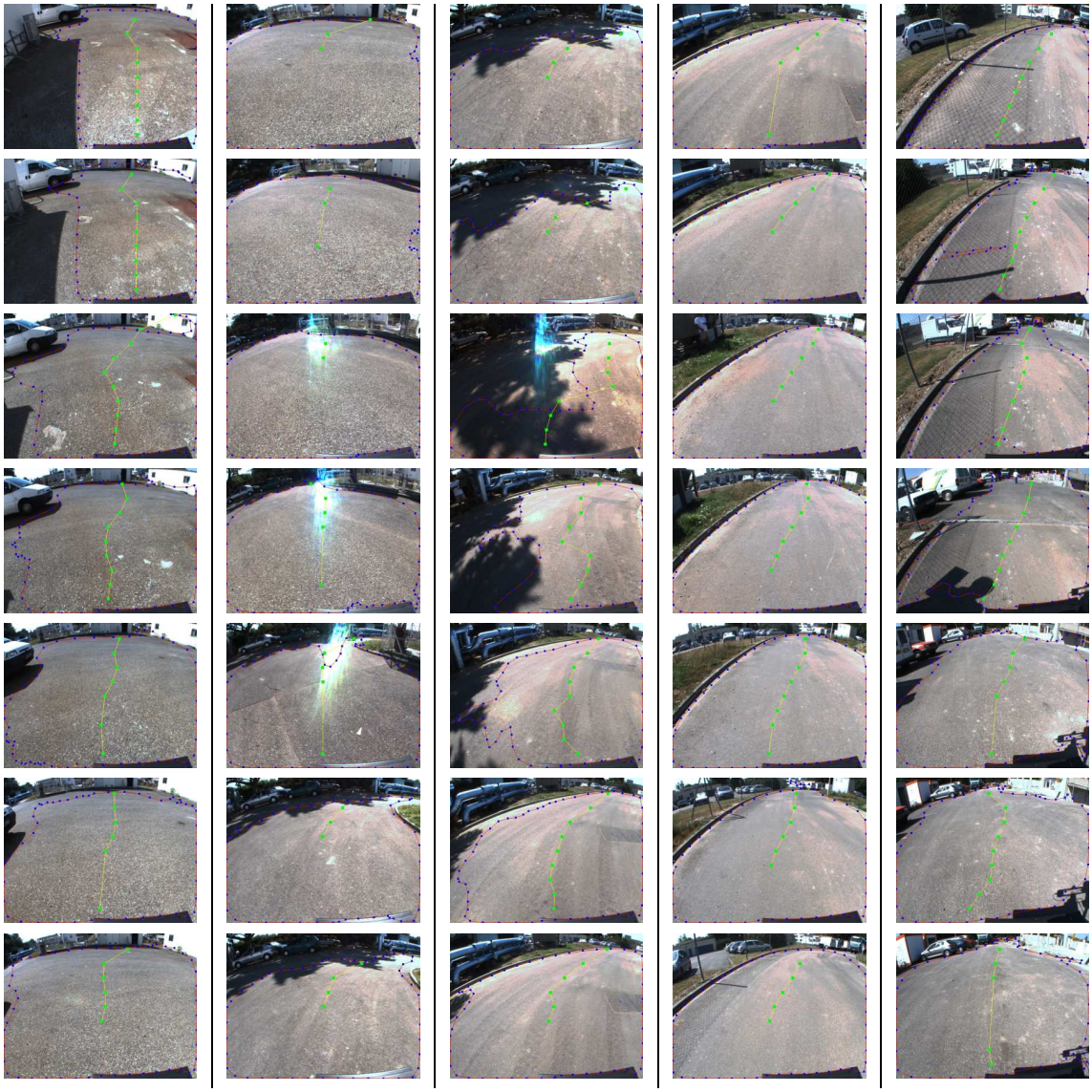


FIG. 5.19 – Séquence montrant le suivi d'un chemin goudronné large

$1 \sim 2\,s$ sur une machine dédiée, le temps exact dépend de la taille des objets dans l´image (plus les objets sont grands, moins de régions sont extraites). Sur le robot le temps est plus incertain pour le partage des ressources.

Une stratégie permettant d'augmenter le champ de vue du robot devra être envisagée pour naviguer correctement sur chemin de largeur considérable. Nous supposons ainsi qu'une caméra active contrôlée astucieusement en orientation et focale, pourra fournir des vues plus adaptées pour la navigation sur chemins.

5.7 Conclusion

Ce chapitre a présenté une application en robotique, de méthodes conçues pour l'extraction et le suivi de chemins à partir d'images fournies par une caméra couleur. Afin de mettre en oeuvre des techniques de navigation visuelle, nous avons proposé plusieurs primitivesélémen-

(a) Erreurs de segmentation et classification des régions lointaines

(b) Perturbation du contours par ombrages

(c) Fausse trajectoire (fort ombrage)

FIG. 5.20 – Erreurs récurrentes lors de l'interprétation visuelle d'images

taires de déplacement. Ces primitives de base sont censées profiter des caractéristiques naturelles de l'environnement semi-structuré agricole pour réaliser des mouvements élémentaires du type « Suivre un chemin ». L´architecture utilisée permet l´intégration de nouveaux comportements et son utilisation par un planificateur-superviseur de haut niveau.

Le système de vision présente certains avantages clairs tels que l'initialisation automatique du suivi aussi bien que les mises à jour périodiques (re-initialisations) du *tracking*, ce qui permet de corriger des possibles erreurs dues à une dérive temporelle ou à des variations des conditions d'illumination. L'identification correcte d'autres classes d'objets que le chemin, est une propriété pertinente de la description 2D de la scène qui peut servir pour l'optimisation d'un processus de détection d'obstacles focalisé uniquement sur ces zones.

Cependant, l'information stéréo $3D$ doit être inclue dans notre système pour l'évitement d'obstacles. De plus, cette information $3D$ nous fournirait une approximation plus fidèle du sol pour travailler sur des terrains plus accidentés où l'hypothèse de sol plat n'est plus valable. D'après nos expérimentations en terrains peu accidentés, nous avons constaté la validité de l'hypothèse de sol plat, même pour de longues distances.

Nos expériences préliminaires montrent des résultats très prometteurs ; néanmoins certaines difficultés doivent faire encore l'objet de la recherche à venir. Par exemple, la classification des objets en utilisant la couleur et la texture doit être plus robuste aux diverses conditions de l'environnement, et doit être renforcée avec d'autres attributs explicites tels que la forme ou la situation dans l'image.

Notons également, que nous avons atteint un taux de reconnaissance global de 90% de bonne identification, et l'interprétation complète de la scène est produite en environ 1 s. À cet effet, il faudrait réaliser un traitement spécifique afin de pallier ou de corriger les difficultés provoquées par les ombres dans les images. Évidemment, la seule utilisation de l'extraction du chemin, n'est pas suffisante pour satisfaire les contraintes temps réel pour certains applications. Cependant, l'exploitation d'une fonction de suivi visuel, fondée sur des contours actifs nous a permis de travailler en temps réel. Cet algorithme fonctionne largement plus rapidement que l'algorithme d'extraction de route ou de chemin, mais la coopération entre ces deux modalités visuelles devrait faire l'objet de travaux complémentaires, car une synchronisation adéquate entre les deux fonctions s'avère cruciale.

L'asservissement visuel 3D (*position-based*) nous a permis d'obtenir de bons résultats ; son principal désavantage est l'absence de critères pour maintenir la cible dans le champ de vue.

Ainsi, une comparaison entre les résultats obtenus et ceux provenant d'un asservissement visuel classique 2D (*image-based*) pour la primitive de suivi d´objets serait intéressante.

Conclusion générale

Ce mémoire de thèse propose une contribution au problème de la navigation visuelle d'un robot dans des environnements semi-structurés d'extérieur. Nous proposons une méthode de navigation qui exploite uniquement une caméra couleur portée sur un robot mobile. Les domaines d'application de ce travail se situent dans l'automatisation de machines agricoles, en vue de la navigation automatique dans un réseau de chemins (pour aller d'une ferme à un champ labouré par exemple), et de la construction et la mise à jour d'une représentation de ce réseau.

L'approche que nous avons développée, se fonde sur la construction d'une description $2D$ de la scène perçue à partir de la segmentation en régions de l'image couleur, puis de la caractérisation et de la classification des régions segmentées par des attributs de couleur, de texture et par des informations contextuelles. L'identification des régions correspondant aux zones navigables, et en particulier aux chemins, s'avère suffisante pour traiter ensuite de la navigation du robot sur un chemin supposé plat, en calculant une trajectoire sur le sol à partir des contours des régions identifiées *Chemin* dans l'image, puis en exécutant des mouvements asservis sur ces trajectoires.

Les méthodes ont été implémentées et mises en oeuvre sur le démonstrateur Dala du groupe *R*obotique et Intelligence Artificielle du LAAS-CNRS. Notre travail a permis l'intégration de plusieurs modules de perception et de commande dans le système embarqué sur ce robot. En particulier, nous avons intégré sur Dala, et validé par plusieurs expérimentations, des commandes référencées Vision, telles que « Suivre un chemin », « Suivre une bordure », ou « Aller vers objet ». Dans le but d'augmenter la fréquence de l'envoi des commandes vers le module de locomotion, nous avons exploité une fonction de suivi visuel des coutours du chemin par une méthode de contours actifs (ou *Snakes*), adaptée pour exploiter la couleur.

Le chapitre 2 a présenté une chaîne de pré-traitement pour le demosaïquage, le calibrage chromatique (balance de blancs) et le rendu correct d'images couleur acquises depuis la caméra mono-CCD doté d'un filtre *Bayer* montée sur le robot. Nous avons introduit dans ce chapitre les modèles d'adaptation chromatique qui sont très peu utilisés pour la navigation visuelle de robots. [Bianco 02] est la seule application robotique qui, à notre connaissance, a exploité un modèle d'adaptation chromatique (*retinex*) pour atténuer les effets négatifs des changements d'illumination.

Le chapitre 3 a décrit en détail la procédure monoculaire permettant d'obtenir une description $2D$ de la scène perçue depuis la caméra couleur. Pour cela, une technique hybride de segmentation couleur (seuillage chromatique/croissance des régions) nous permet d'extraire les principales régions de l'image : des attributs de couleur, de texture (histogrammes de sommes et différences) et de contexte sont ensuite calculés sur chacune des régions segmentées. En phase d'apprentissage, une base de connaissances est construite en mode supervisé, afin de mémoriser

les attributs caractéristiques de régions correspondant à des zones *Arbre*, *Ciel*, *Champ*, *Herbe*, *Chemin*, *etc.* de la scène. Puis en phase d'identification, les régions caractérisées par ce même vecteur d'attributs sont classifiées en exploitant une méthode de type *Support Vector Machines* (SVM) ou K-Plus Proches Voisins (k-PPV) . Les régions bien classifiées sont fusionnées aux régions voisines (connexes) appartenant à la même classe, ce qui permet de corriger des erreurs liées à la sur-segmentation initiale de l'image.

Nous avons décrit une approche pour la classification des régions, composée d'une étape préalable de pré-traitement et apprentissage de données. Une contribution à ce travail a été l'implémentation d'un algorithme de pré-traitement de données par une Analyse en Composantes Indépendantes (ACI) qui s'avère plus puissante que la classique Analyse en Composantes Principales (ACP) . Ces pré-traitements ont permis d'améliorer le taux global de reconnaissance des régions.

Ainsi est obtenue la description $2D$ de la scène sur laquelle se fonde le module de navigation visuelle. Notons que tous les traitements proposés pour générer cette description, fonctionnent également pour des images obtenues par des capteurs différents (tri-CDD, appareils de photographie numérique et images de synthèse) : la base de données du module de reconnaissance peut être réutilisée dans la plupart des cas.

Dans un réseau de chemins, le robot doit reconnaître les intersections afin de naviguer et de construire un modèle topologique de son parcours. Dans le chapitre 4, l'extraction des régions navigables (*chemins*) est d'abord présentée : une approche fondée sur la représentation des contours appelée *Shape Context*, a été proposée pour décrire la forme des chemins et pour la catégorisation des configurations de chemins (ligne droite, virage, carrefours...). La reconnaissance à l'aide de *Shape Context*, permet la détection des intersections inter-chemins, exploitées soit pour la construction de cartes topologiques, soit pour la navigation en identifiant les configurations caractéristiques de chemins. Les résultats dans la catégorisation des chemins sont encourageants malgré les formes très variables d'intersections que nous avons trouvées dans la nature. Cependant, il faudra valider cette méthode sur un réseau de chemins réel, et l'intégrer avec une fonction de localisation qualitative afin de détecter quand le robot repasse sur une intersection déjà explorée.

Enfin, dans la dernière partie de la thèse, nous décrivons le module de navigation visuelle mis en oeuvre sur notre démonstrateur Dala, et les résultats expérimentaux. Ce module exploite les descriptions $2D$ pour guider les déplacements du robot sur les zones navigables, et en particulier sur les chemins. Pour cela, nous avons implémenté sur le robot, plusieurs primitives de déplacement élémentaires (*Suivre un chemin, Aller vers objet, Suivre une bordure,...*) ; un planificateur pourra exploiter ces primitives pour générer des déplacements plus complexes, sous la forme d'une séquence de primitives élémentaires.

Les contours échantillonnés et filtrés des régions *Chemin*, sont modélisés par des *splines* cubiques nous permettant d'estimer rapidement dans l'image, une trajectoire à suivre. Une approximation plus régulière des points formant cette trajectoire est obtenue en utilisant une courbe de Bézier. En utilisant l'hypothèse de sol plat, cette trajectoire est projetée sur le sol ; les éléments de trajectoire ainsi générés depuis chaque image, sont fusionnés, filtrés et traduits en commande pour le robot. Cette capacité à exploiter toutes les images d'une séquence d'images afin de planifier une trajectoire, rend le système plus robuste.

Nous avons obtenu des résultats expérimentaux satisfaisants en ce qui concerne l'exécution des primitives « Suivre un chemin » et « Suivre une bordure ». Mais les mouvements du robot sont très lents et peu optimaux ; malgré les filtrages et lissages mis en place, l'instabilité sur la

détection des contours des régions ne permet pas toujours de générer des mouvements souples pour le robot.

Le modèle sémantique de la scène est produit à basse fréquence (de 0,5 à 1 Hz) par le module de vision couleur ; nous avons intégré avec celui-ci, un module de suivi temporel des contours du chemin, ce qui permet d'augmenter la fréquence d'envoi des consignes (10 Hz environ) au module de locomotion. Le suivi sur chaque image se fonde sur la méthode des contours actifs utilisant un gradient couleur focalisée ; les bords du chemin sont extraits en exploitant l'information chromatique fournie par le module de vision couleur. Modules de vision couleur et de suivi temporel doivent être synchronisés de sorte que le suivi puisse être réinitialisé en cas de dérive. Plusieurs problèmes liés à cette intégration de deux processus visuels asynchrones, ont été présentés, mais le temps a manqué pour leur donner un début de solution fiable.

Nous considérons qu'une étape de validation dans un vrai milieu rural nous permettrait de mieux détecter les limitations de notre système (vibrations, chemins ombragés, variations d'illumination, portée du système de vision *etc.*), et de proposer des corrections pertinentes.

Perspectives

Enfin, nous souhaitons terminer en évoquant des travaux que nous pourrions mener dans le futur sur la thématique présentée dans cette thèse. Tout d'abord, une première perspective est d'élargir notre domaine d'application en utilisant la vision stéréo, ce qui permettra

- d'améliorer les taux de reconnaissance des régions obtenus en vision monoculaire. Nous avons présenté plusieurs situations pour lesquelles nous ne pensons pas possible de décrire correctement la scène en exploitant uniquement une caméra. Des attributs 3D seront nécessaires pour différencier de manière fiable les classes *Herbe* et *Arbre*, *Chemin* et *Champ* ....
- de naviguer dans des terrains beaucoup plus accidentés et parsemés d'obstacles, donc sans considérer l'hypothèse de sol plat que nous avons exploitée dans ce travail.

Nous avons proposé un schéma de coopération entre les fonctions réalisant l'*extraction du chemin* (R-LOC $\approx$1 Hz) et le *Tracking du chemin*(R-TRACK $\approx$10 Hz) ; nous souhaitons étudier en profondeur cette coopération entre processus visuels pour exploiter au maximum les aspects temporels (ou filtrage) de ces deux tâches tournant en parallèle et qui ont des cadences très différentes.

Dans l'étape de perception, il faudrait réaliser un traitement spécifique afin de reconnaître et corriger les difficultés posées par l'analyse des terrains comportant de forts contrastes entre zones ombragées et ensoleillées. En dehors de l'exploitation de la vision stéréo, ce problème de vision purement monoculaire, nous paraît un défi pour tester les méthodes locales d'adaptation chromatique ou de contraste.

En ce qui concerne le travail passionnant sur le rendu et la reproduction d'images couleur acquises à partir des caméras munies d'un filtre *Bayer*, il est important de perfectionner les algorithmes de balance des blancs qui présentent encore certaines défaillances (parfois imperceptibles) dans les régions saturées des images à cause d'une illumination trop intense. Evidemment dans ces régions, la correction ne doit pas être la même que pour le reste de l'image. En effet, une adaptation souhaitable de notre méthode de balance de blancs, consisterait à pouvoir corriger

localement des images , pour supprimer des *voiles* créés par plusieurs types d'illuminant.

Nous avons en particulier détecté ce problème dans des images acquises à l'intérieur de la Cité de l'Espace à Toulouse où le robot-guide RACKHAM doit traiter des images avec des conditions d'illumination très perturbantes, provoquées principalement par des lumières néon giratoires. De manière moins anecdotique, une étude approfondie pour établir (clairement) les liens entre les modèles d'adaptation chromatique et la constance de couleur doit être menée pour améliorer encore les qualités des images couleur acquises en milieu naturel ; la prise-en-compte de la position de l'illuminant principal, le soleil, devrait permettre de corriger des éblouissements, ou tout au moins, d'invalider les résultats de la vision si nécessaire.

Notre travail se fonde sur une segmentation couleur ; si la couleur disparaît (tombée de la nuit, mauvaises conditions climatiques ou saison automnale...), la segmentation échoue et toute notre approche avec ! Nous envisageons donc la mise en oeuvre d'une technique de segmentation travaillant avec des attributs additionnels. Les inconvénients actuels pour l'utilisation directe de la texture dans les processus de segmentation, sont un temps de calcul important et la mauvaise qualité des frontières obtenues ; mais une approche hybride de segmentation utilisant en parallèle plusieurs composantes (peut-être dans un espace $\mathbb{R}^5$) chromatiques et des descripteurs rapides de texture tel que les codages CCR, LBP ou Census deviendrait très utile quand la couleur des images (saturation) n'est plus discriminante.

Nous considérons que même si la technique de classification des *Support Vector Machines* est robuste et fiable, la réputation croissante des techniques du type *Adaptive Boosting* va nous obliger à vérifier et comparer à court terme leurs avantages et inconvénients pour notre problème de classification. En outre, il est nécessaire de reprendre l'idée de R.Murrieta sur une pré-classification, qui consiste à disposer d'une méthode de détection automatique du contexte (temps, saison ...) afin de sélectionner la base de données la plus appropriée pour l'extraction correcte de la région navigable ou de la cible à suivre.

Enfin, évoquons le problème certainement le plus difficile, sur lequel nos résultats actuels restent très insuffisants : la catégorisation des chemins, afin de reconnaître les formes de chemins (ligne droite, virages) et surtout, les types d'intersection. La méthode fondée sur les descripteurs « Shape Context », a besoin d'être validée sur un terrain expérimental ayant les éléments appropriés (réseau de chemins, *etc.*). Ce descripteur « Shape Context » doit être enrichi avec d'autres attributs afin de le rendre plus robuste dans la détection des chemins réels et surtout parce que les histogrammes correspondants ne seront pas toujours discriminants (nombre important de *bins* nuls).

La vision panoramique (omnidirectionelle) doit être prise en compte car elle va nous fournir une signature globale des lieux : amers naturels rencontrés lors du parcours du robot, signature globale des intersections... Les méthodes fondées sur l'apparence développées au LAAS par J.Gonzalez [Gonzalez-Barbosa 04] devraient être intégrées pour traiter de la localisation qualitative du robot dans le modèle topologique du réseau de chemins. Cette technique permettrait de limiter le besoin d'une localisation métrique précise : notre machine pourrait se contenter d'une localisation GPS standard.

Enfin, la reproduction de ces expériences sur une machine agricole sur des sols mexicains (généralement plus arides), est prévue à court terme dans le cadre d'un projet de collaboration Franco-Mexicain, financé par le laboratoire LAFMI, projet entre le LAASet l'Université de Guanajuato.

Annexe A

Représentation de la couleur

La couleur est un attribut visuel perçu par les humains comme une combinaison tri-stimuli (R, G, B) : ces trois composantes sont appelées les couleurs primaires. Les composantes tri-stimuli (R,G,B) sont représentées par les valeurs de luminosité sur la scène, elles sont obtenues en utilisant trois filtres séparés et centrés sur différentes longueurs d'onde λ (425.8 nm pour le bleu, 546.1 nm pour le vert et 650.0 nm pour le rouge),

$$R = \int_{\lambda} E(\lambda) S_R(\lambda)\, d\lambda \qquad G = \int_{\lambda} E(\lambda) S_G(\lambda)\, d\lambda \qquad B = \int_{\lambda} E(\lambda) S_B(\lambda)\, d\lambda\,, \tag{A.1}$$

où S_R, S_G et S_B sont les filtres colorés placés devant le spectre de la lumière d'incidence (*radiance*). Pour des opérations d'affichage, cet espace est très utilisé (écrans d'ordinateur, des systèmes de télévision, appareils photo et caméras numériques) ; en revanche, la corrélation entre ses composantes chromatiques le rend inadéquat à la segmentation couleur. Une corrélation implique une interdépendance entre les changements d'intensité et les variations de chacune des composantes chromatiques ; la représentation RGB est donc très sensible aux changements d'illumination [Celenk 95]. Ainsi, la distance entre couleurs dans cet espace ne représente pas une mesure perceptuellement uniforme des différences de couleur.

Il existe donc de nombreuses autres représentations de la couleur utilisées en traitement d'images (HSI, $I_1I_2I_3$, $CIE\ L^*u^*v^*$, $CIE\ L^*ab$, ...) ; il n'existe pas de consensus dans la communauté scientifique, sur la *meilleure* représentation qui se révélerait supérieure en performance, surtout pour coder la couleur dans des images acquises sur des scènes réelles d'extérieur. La sélection de l'espace de couleur est donc le premier choix important à prendre en compte lors de la mise en oeuvre d'une technique de segmentation.

Les représentations de la couleur sont classées en deux groupes : les espaces de couleur linéaires et non-linéaires.

Les principales représentations linéaires de la couleur sont :

– le système XYZ Le passage des coordonnées RGB vers les coordonnées primaires XYZ, dites virtuelles[35], s'obtient grâce à la matrice suivante :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.488718 & 0.310680 & 0.200602 \\ 0.176204 & 0.812985 & 0.010811 \\ 0.000000 & 0.010205 & 0.989795 \end{pmatrix} \begin{pmatrix} R \\ G \\ B\,. \end{pmatrix} \tag{A.2}$$

[35] Y représente approximativement la sensibilité de l'oeil humain à la luminosité (considérée comme la composante de luminance du spectre incident)

– le système *YIQ* est utilisé pour codifier l'information couleur sur les systèmes américains des signaux de télévision (NTSC),

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.253 & -0.312 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix}, \tag{A.3}$$

où $0 \leq r, g, b \leq 1$, sont les composantes RGB normalisées :

$$r = \frac{R}{R+G+B}; g = \frac{G}{R+G+B}; b = \frac{B}{R+G+B}; \tag{A.4}$$

La composante Y représente la luminance tandis que I et Q sont respectivement les composantes chromatiques représentant les oppositions cyan-orange et magenta-bleu.

– l'espace YUV est le standard adopté pour les systèmes européens de télévision (PAL),

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix}, \tag{A.5}$$

où $0 \leq r, g, b \leq 1$. Les composantes YUV ont la même signification que le standard YIQ mais la particularité de cet espace consiste à utiliser le blanc de référence D_{65}.

– la représentation appelée *YES* a été conçue par la SMPTE[36] et utilisée par *XEROX*.

$$\begin{pmatrix} Y \\ E \\ S \end{pmatrix} = \begin{pmatrix} 0.253 & 0.684 & 0.063 \\ 0.500 & -0.500 & 0.000 \\ 0.250 & 0.250 & -0.500 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{A.6}$$

Elle contient la luminance Y, les composantes de chrominance E (l'axe rouge-vert) et S (l'axe jaune-bleu).

– l'espace $I_1I_2I_3$ développé par Ohta [Ohta 80, Ohta 85] est le résultat d'une expérimentation avec une centaine d'attributs couleur qui ont été utilisés pour évaluer une méthode de segmentation (seuillage récursif [Ohlander 78]) sur différents types d'images. Cet espace provient d'une transformation de Karhunen-Loéve pour déterminer les attributs couleur non corrélés (orthogonaux) et les plus discriminants,

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & -1/2 \\ -1/4 & 1/2 & -1/4 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \tag{A.7}$$

où I_1 correspond à la composante de luminance (intensité). I_2 et I_3 sont respectivement les composantes couleur en opposition bleu-rouge et magenta-vert. Plusieurs expériences [Ohta 80, Lee 94, Murrieta-Cid 98] sur la segmentation couleur en comparant plusieurs espaces de couleur ont proclamé que l'espace d'Ohta manifeste le meilleur compromis entre la qualité de segmentation et la complexité algorithmique.

[36]Society of Motion Picture and Television Engineers

Les transformations non-linéaires de couleur sont plus souvent utilisées pour les applications de traitement d'images que les transformations linéaires. Ceci s'explique par plusieurs raisons : elles sont parfois plus robustes aux changements d'illumination et plusieurs d'entre elles se sont inspirées de la perception visuelle de la couleur chez l'être humain. Les espaces de couleur non-linéaires les plus utilisés sont :

– les applications réelles ont besoin d'un espace bien indépendant aux changements d'éclairage. L'espace normalisé *rgb* a été conçu pour obtenir des variations d'intensité uniformes sur toute la distribution spectrale de couleur. Les composantes chromatiques normalisées L_1 sont définies de la manière suivante :

$$(r, g, b) = \frac{(R, G, B)}{R + G + B}\,, \tag{A.8}$$

Comme $r+g+b = 1$, seulement deux composantes sur trois sont linéairement indépendantes et vraiment utiles. Cette transformation est caractérisée par une certaine robustesse aux variations d'illumination et par la singularité à l'origine qui provoque des irrégularités dans la couleur à basse intensité.
Il est possible de définir un espace normalisé [Healey 92] muni d'une différenciation plus uniforme entre les propriétés chromatiques, en utilisant à la place de la norme L_1, la norme L_2 définie comme suit,

$$(r, g, b) = \frac{(R, G, B)}{\sqrt{R^2 + G^2 + B^2}}\,, \tag{A.9}$$

– la représentation $l_1 l_2 l_3$, proposée par T. Gevers [Gevers 99], a trouvé des applications dans la reconnaissance d'objets grâce à une certaine robustesse aux changements d'illumination. Elle est définie de la manière suivante :

$$(l_1, l_2, l_3) = \frac{(|r-g|, |r-b|, |g-b|)}{|r-g| + |r-b| + |g-b|}\,, \tag{A.10}$$

– l'espace de couleur *CIE-L*ab* fut développé pour représenter une uniformité perceptuelle de la couleur compatible avec la notion psychophysique de la couleur chez l'humain. Il essaye de prendre en compte la réponse logarithmique de l'oeil ; de ce fait, il est très utilisé dans le cas de mélanges de pigments (industrie graphique, photographie, restauration de peintures,...). Nous utilisons cet espace pour corriger en ligne les déséquilibres chromatiques, provoqués en particulier par les variations d'illumination (voir *cf.* § 2.4.3.4).
La luminosité $L*$ et les composantes chromatiques a et b sont données par les équations suivantes, en fonction des coordonnées primaires XYZ (calculées par une transformation linéaire en fonction de RGB - voir ci-dessus l'équation A.2) :

$$L^* = \begin{cases} 116\left(\frac{Y}{Y_w}\right)^{\frac{1}{3}} - 16 & \text{si } \frac{Y}{Y_w} > 0.008856 \\ 903.292\,\frac{Y}{Y_w} & \text{si } \frac{Y}{Y_w} \leq 0.008856, \end{cases} \quad \begin{aligned} a &= 500\left[f\left(\frac{X}{X_w}\right) - f\left(\frac{Y}{Y_w}\right)\right], \\ b &= 200\left[f\left(\frac{Y}{Y_w}\right) - f\left(\frac{Z}{Z_w}\right)\right] \end{aligned} \tag{A.11}$$

où X_w, Y_w et Z_w symbolisent le blanc de référence qui dans notre application a été fixé à (255, 255, 255). Notons que la fonction $f(t)$ va gérer la stabilité (ou le bruit) de cet espace à basse intensité ; elle est définie comme suit :

$$f(t) = \begin{cases} t^{\frac{1}{3}} & \text{si } t > 0.008856 \\ 7.787\,t + \frac{16}{116} & \text{si } t \leq 0.008856 \end{cases} \tag{A.12}$$

La racine cubique utilisée dans ces équations est très intéressante car des expériences psycho-visuelles ont montré un comportement non-linéaire analogue à celui de l'oeil humain. De plus, l'introduction du rapport $\frac{Y}{Y_w}$ permet de simuler grossièrement l'adaptation de l'oeil humain à une luminosité de référence.
La luminance L^* donne la notion de clarté tandis que la chrominance est représentée par a et b qui sont respectivement l'opposition de couleur vert-rouge et l'opposition bleu-jaune. La notion d'uniformité est mieux définie que dans l'espace RGB : la distance euclidienne entre deux couleurs, ou écart visuel, devrait être perçue par l'oeil humain avec la grandeur $\Delta E_{ab} = \sqrt{\Delta L^{*2} + \Delta a^2 + \Delta b^2}$.
Il est possible de définir la notion de teinte et de saturation à partir des composantes a et b. La teinte est définie par la valeur angulaire : $h = tan^{-1}(b/a)$ et la saturation ou chroma (niveau de coloration) par la grandeur $C = \sqrt{a^2 + b^2}$ qui représente la distance euclidienne à l'axe achrome.

– l'espace perceptuel *CIE-*L^*uv est défini de façon similaire à l'espace L^*ab, en particulier le calcul de la luminance L^* utilise la même expression (*cf.* équations A.2 et A.11) pour ces deux représentations. Par contre, les composantes de chrominance sont données par :

$$u = 13\, L^*(u' - u_w)\,, \qquad v = 13\, L^*(v' - v_w)\,, \tag{A.13}$$

$$u' = \frac{4X}{X + 15Y + 3Z}\,, \qquad v' = \frac{6Y}{X + 15Y + 3Z}\,, \tag{A.14}$$

où u_w, v_w sont les valeurs correspondant au blanc de référence (X_w, Y_w, Z_w).
De la même manière que pour l'espace L^*ab les définitions de teinte et le chroma peuvent être obtenues par $h = tan^{-1}(u/v)$ et $C = \sqrt{u^2 + v^2}$. De la même manière, les différences en couleur pour cet espace sont calculées par $\Delta E_{uv} = \sqrt{\Delta L^{*2} + \Delta u^2 + \Delta v^2}$.
Ces deux derniers espaces sont particulièrement efficaces pour mesurer de petites variations de la couleur mais les problèmes de singularité sont encore l'un des points faibles de ces approximations.

– l'espace *ITS* est un espace qui reste très utilisé dans des applications de traitement d'images. Il correspond davantage à la notion habituelle de couleur chez l'homme, où les variations de saturation sont facilement détectées. Malheureusement il n'existe pas une formulation mathématique unique et plusieurs variantes peuvent donc être disponibles.
I correspond à la valeur d'intensité (quantité de lumière) tandis que l'information de couleur est donnée par la teinte T (longueur d'onde dominante) et la saturation S (pureté de la couleur). La teinte est calculée, comme dans le cas des espaces perceptuels (L^*ab et L^*uv), par un angle entre une ligne de référence et un point RGB. La saturation représente la distance minimale à l'axe achrome. La transformation la plus courante pour calculer les composantes ITS à partir des composantes tri-stimuli RGB est donnée par les équations suivantes :

$$I = \frac{R + G + B}{3}\,, \quad T = tan^{-1}\left(\frac{\sqrt{3}(G - B)}{(R - G) + (R - B)}\right)\,, \quad S = 1 - \frac{min(R, G, B)}{I} \tag{A.15}$$

Comme pour la plupart des représentations non-linéaires, cet espace a une singularité non amovible sur l'axe achrome qui provoque une forte instabilité dans la teinte. En pratique les valeurs de faible saturation doivent être redéfinies ou ne doivent pas être prises en compte.

Glossaire

Nous donnons ici plusieurs acronymes et quelques définitions qui interviennent fréquemment dans ce mémoire. Nous avons essayé de les définir dans la plupart des cas lors de leur première utilisation, sauf pour les notions les plus courantes.

Ada Boost : Adaptive boosting
AI : (IA en français) Artificial Intelligence

Bâtonnets : Cellules nerveuses de la rétine, sensible à la lumière par son prolongement qui contient le pourpre rétinien, sont à l'origine de la vision périphérique et de la vision crépusculaire. Ils sont beaucoup plus sensibles à la lumière que les cônes, mais ils fournissent des images floues et incolores
BWGM : bipartite weighted graph matching

Cônes : Cellules photosensibles de la rétine, responsables de la perception colorée lorsque l'éclairage est suffisant (vision diurne ou photopique). La rétine contient trois sortes de cônes, dont le maximum d'absorption se situent dans le bleu, le vert ou le rouge.
Capteur Super CCD : Le super CCD est composé de photodiodes octogonales disposées en nid d'abeilles alors que dans un capteur mono CCD les photodiodes sont rectangulaires. Ce procédé permet théoriquement de loger plus de photosites sur une même surface et donc d'obtenir plus de pixels et des photos d'une meilleure qualité que celles faites grâce à un appareil équipé d'un capteur (mono) CCD.
CCD : Charge Coupled Device
CCDA : Combined Constraint Data Association
CFA : Color Filter mosaic Array
CIE : Commission International de l'Eclairage
CMOS : Complementary Metal Oxide Semiconductor

EKF : Extended Kalman Filter

Gammut : Espace couleur relatif à un périphérique ou un système donné, délimitant l'ensemble des couleurs reproductibles par le périphérique ou le système
GPS : Global Positioning System

ICA : Independent Component Analysis
INRIA : Institut National de la Recherche en Informatique et Automatique
Intensité lumineuse : Energie lumineuse émise par une source dans un angle solide, s'exprime en candela.

LASMEA : Le Laboratoire des Sciences et Matériaux pour l'électronique et l'Automatique
LDA : Linear Discriminant Analysis
LIP6 : Laboratoire d'Informatique de l'Université Paris 6
Luminance : Quotient de l'intensité lumineuse par l'aire apparente de la surface émissive. La luminance s'exprime en candela par mètre carré
Luminosité : Perception visuelle de la luminance. La luminosité fait référence à la quantité de lumière qui est absorbée ou réfléchie par la zone colorée

MLP : Multi Layer Perceptron
MRF : Markov Random Fields

OECF : Opto-Electronic Conversion Function

PCA : Principal Component Analysis
Plage dynamique : La plage dynamique caractérise l'étendue des tonalités, des zones les plus sombres aux zones les plus claires, qu'un appareil est capable de capturer sans perte de détails

QP : Quadratic Programming

RAG : Region Adjacency Graph
RBF : Radial Basis Function

SLAM : Simultaneous Localisation And Mapping
SMPTE : Society of Motion Picture and Television Engineers
STFT : Short-Time Fourier Transform
SVD : Singular Value Decomposition
SVM : Support Vector Machines

Teinte : L'aspect de la couleur qui se distingue par la longueur des ondes lumineuses et varie selon les longueurs d'ondes.
tessellation : le recouvrement d'une surface ou région à l'aide de polygones placés de façon à ne laisser aucun espace ou n'avoir aucune superposition entre les polygones. Ce terme est connu aussi comme carrellage

Index

Cet index contient des références vers un ensemble de mots clefs de ce document

Références bibliographiques

[Adams Jr. 97] James E. Adams Jr. & John F. Hamilton Jr. Adaptive color plan interpolation in single sensor color electronic camera, July 29 1997. United States Patent 5,652,621.

[Alami 98] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An Architecture for Autonomy*, 1998.

[Alleysson 01] David Alleysson & Jeanny Hérault. *Interpolation d'images couleurs sous échantillonnées par un modèle de perception.* In Groupe de Recherche sur le Traitement du Signal et des Images (GRETSI), Toulouse, France, septembre 2001.

[Alleysson 02] David Alleysson, Sabine Süsstrunk & Jeanny Hérault. *Color Demosaicing by Estimating Luminance and Opponent Chromatic Signals in the Fourier Domain.* In The 10^{th} Color Imaging Conference (CIC), pages 331–336, Scottsdale, Arizona, USA, November 12 2002.

[Altunbasak 98] Yucel Altunbasak, P. Erhan Eren & A. Murat Tekalp. *Region-Based Parametric Motion Segmentation Using Color Information.* Graphical Models and Image Processing, vol. 60, no. 1, pages 13–23, January 1998.

[Amini 90] A. A. Amini, T. E. Weymouth & R. C. Jain. *Using Dynamic Programming for Solving Variational Problems in Vision.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 9, pages 855–867, September 1990.

[Antani 02] Sameer Antani, Rangachar Kasturi & Ramesh Jain. *A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video.* Pattern Recognition, vol. 35, no. 4, pages 945–965, April 2002.

[Antonini 03] Gianluca Antonini, Vlad Popovici & Jean-Philippe Thiran. *Independent Component Analysis and Support Vector Machine for Face Feature Extraction.* In Intl. Conf. on Audio- and Video-Based Biometrie Person Authentication (AVBPA), volume LNSC 2688 of *Lecture Notes in Computer Science*, pages 111–118, Guildford, UK, June 9-11 2003. Springer-Verlag.

[Athitsos 04] Vassilis Athitsos & Stan Sclaroff. *Boosting Nearest Neighbor Classifiers for Multiclass Recognition.* Rapport de recherche 2004-006, Boston University, Computer Science, February 5 2004.

[Aufrere 03] R. Aufrere, J. Gowdy, C. Mertz, C. Thorpe, C. Wang & T. Yata. *Perception for collision avoidance and autonomous driving.* Mechatronics, vol. 13, no. 10, pages 1149–1161, December 2003.

[Aufrere 04] R. Aufrere, V. Marion, J. Laneurit, C. Lewandowski, J. Morillon & R. Chapuis. *Road sides recognition in non-structured environments by vision.* In IEEE Intelligent Vehicles Symposium, pages 329–334, Parma, Italy, June 2004.

[Aufrère 00] Romuald Aufrère, Roland Chapuis & Frédéric Chausse. *A fast and robust vision based road following algorithm.* In IEEE intelligent Vehicles Symposium, Dearborn, Michigan (USA), October 3-5 2000.

[Aufrère 01] Romuald Aufrère, Roland Chapuis & Frédéric Chausse. *A model-driven approach for real-time road recognition.* Machine Vision and Applications, Springer-Verlag, vol. 13, no. 2, 95–107 2001.

[Autio 03] Ilkaa Autio & Tapio Elomaa. *Flexible view recognition for indoor navigation based on Gabor filters and support vector machines.* Pattern recognition, vol. 36, no. 12, pages 2769–2799, December 2003.

[Avina-Cervantes 02] Gabriel Avina-Cervantes, Michel Devy & R. Murrieta-Cid. *Road detection for robot navigation.* In 3rd International Symposium on Robotics and Automation, ISRA 2002, volume 1, Toluca (Mexico), Septembre 1-4 2002.

[Avina-Cervantes 03a] G. Avina-Cervantes, M. Devy & A. Marin-Hernandez. *Lane extraction and tracking for robot navigation in agricultural applications.* In IEEE The 11th International Conference on Advanced Robotics, volume 2, pages 816–821, Coimbra, Portugal, June 30 - July 3 2003.

[Aviña-Cervantes 03b] Gabriel Aviña-Cervantes & Michel Devy. *Détection et suivi de chemins pour la navigation d'un robot en milieu naturel.* In Journées Francophones des Jeunes Chercheurs en Vision par Ordinateur (ORASIS'2003), volume 1, pages 403–410, Gérardmer (France), 19-23 Mai 2003.

[Ayala-Ramirez 00] V. Ayala-Ramirez, J. B. Hayet, F. Lerasle & M. Devy. *Visual localization of a mobile robot in indoor environments using planar landmarks.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2000), pages 275–280, Takamatsu (Japon), November 2000.

[Aycard 47] O. Aycard, F. Charpillet, D. Fohr & J. F. Mari. *Place Learning and Recognition using Hidden Markov Models.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, page 1997, Grenoble, France, September 1741–1747.

[Baluja 96] S. Baluja. *Evolution of an artificial neural network based autonomous land vehicle controller.* IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 26, no. 3, pages 450–463, June 1996.

[Baradez 04] M. O. Baradez, C. P. McGuckin, N. Forraz, R. Pettengell & A. Hoppe. *Robust and automated unimodal histogram thresholding and potential applications.* Pattern Recognition, vol. 37, no. 6, pages 1131–1148, June 2004.

[Baumgartner 99] A. Baumgartner, C. Steger, H. Mayer, W. Eckstein & H. Ebner. *Automatic road extraction in rural areas.* In International Archives of Photogrammetry and Remote Sensing, volume XXXII, Part 3-2W5, pages 107–112, 1999.

[Belkasim 91] S. O. Belkasim, M. Shridhar & M. Ahmadi. *Pattern recognition with moment invariants : A comparative study and new results.* Pattern Recognition, vol. 24, no. 12, pages 1117–1138, December 1991.

[Belluta 00] P. Belluta, R. Manduchi, L. Matthies, K. Owens & A. Rankin. *Terrain Perception for DEMO III.* In IEEE Intelligent Vehicles Symposium 2000, pages 326–332. Dearborn, Michigan, October 2000.

[Belongie 01] S. Belongie, J. Malik & J. Puzicha. *Matching Shapes.* In In Eighth IEEE International Conference on Computer Vision, volume 1, pages 454–461, Vancouver (Canada), July 2001.

[Belongie 02] Serge Belongie, Jitendra Malik & Jan Puzicha. *Shape Matching and Object Recognition Using Shape Context.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pages 509–522, April 2002.

[Belouchrani 97] Adel Belouchrani, Karim Abed-Meraim, Jean-François Cardoso & Eric Moulines. *A Blind Source Separation Technique Using Second-Order Statistics.* IEEE Transactions on Signal Processing, vol. 45, no. 2, pages 434–444, February 1997.

[Betge-Brezetz 96a] S. Betge-Brezetz, P. Hebert, R. Chatila & M. Devy. *Uncertain map making in natural environments.* In IEEE International Conference on Robotics and Automation (ICRA'96), Minneapolis (USA), pages 1048–1053, April 1996.

[Betgé-Brezetz 96b] Stéphane Betgé-Brezetz. *Modélisation incrémentale et localisation par amers pour la navigation d'un robot mobile autonome en environnement naturel.* Thèse de doctorat, Université Paul Sabatier, LAAS/CNRS, Toulouse (France), Février 1996.

[Bianco 02] Giovanni M. Bianco & Alessandro Rizzi. *Chromatic adaptation for robust visual navigation.* Advanced Robotics, vol. 16, no. 3, pages 217–232, May 1 2002.

[Boser 92] Bernhard E. Boser, Isabelle M. Guyon & Vladimir N. Vapnik. *A training Algorithm for Optimal Margin Classifiers.* In IEEE Transactions on Neural Networks, Fifth Annual Workshop on Computational Learning Theory, volume 5, pages 144–152. Pittsbourgh, USA, PA :ACM, July 27-29 1992.

[Brainard 97] D. H. Brainard & W. T. Freeman. *Bayesian color constancy.* Journal of the Optical Society of America A, vol. 14, no. 7, pages 1393–1411, July 1997.

[Brainard 01] D. H. Brainard. *Color vision theory.* In International Encyclopedia of the Social and Behavioral Sciences, vol. 4, pages 2256–2263, 2001.

[Brethes 04] L. Brethes, P. Menezes, F. Lerasle & J. Hayet. *Face tracking and hand gesture recognition for human-robot interaction.* In IEEE International Conference on Robotics and Automation (ICRA), volume 2, pages 1901–1906, New Orleans, LA, USA, April 2004.

[Campbell 97] N. W. Campbell & B. T. Thomas. *Automatic Selection of Gabor Filters for Pixel Classification.* In Sixth International Conference on Image Processing and its Applications, volume 2, pages 761–765, Dublin, Ireland, July 1997.

[Celaya 02] E. Celaya & C. Torras. *Visual Navigation Outdoors : The Argos Project.* In The 7th International Conference on Intelligent Autonomous Systems (IAS-7), pages 63–67, Marina del Rey, California, USA, March 25–27 2002.

[Celenk 90] M. Celenk. *A color clustering technique for image segmentation.* Computer Vision, Graphics and Image Processing, vol. 52, no. 2, pages 145–170, November 1990.

[Celenk 95] M. Celenk. *Analysis of Color Images of Natural Scenes.* Journal of Electronic Imaging (JEI), vol. 4, no. 4, pages 382–396, October 1995.

[Cha 02] Sung-Hyuk Cha & Sargur N. Srihari. *On measuring the distance between histograms.* Pattern Recognition, vol. 35, no. 6, pages 1355–1370, June 2002.

[Chang 99] Ed Chang, Shiufun Cheung & Davis Y. Pan. *Color filter array recovery using a threshold-based variable number of gradients.* In Proceedings of SPIE, volume 3650, pages 36–43, January 27-28 1999.

[Chapuis 95] R. Chapuis, A. Potelle, J. L. Brame & F. Chausse. *Real-Time Vehicle Trajectory Supervision on the Highway.* International Journal of Robotics Research, vol. 14, no. 6, pages 531–542, December 1995.

[Chatila 85] R. Chatila & J. Laumond. *Position referencing and consistent world modeling for mobile robots*. In IEEE International Conference on Robotics and Automation, pages 138–170, 1985.

[Chatila 95a] R. Chatila, S. Lacroix, T. Siméon & M. Herrb. *Planetary exploration by a mobile robot : Mission teleprogramming and autonomous navigation*. Autonomous Robots Journal, vol. 2, no. 4, pages 333–344, 1995.

[Chatila 95b] Raja Chatila. *Autonomous navigation in natural environments*. Robotics and Autonomous Systems, vol. 16, pages 196–211, 1995.

[Chatila 98] R. Chatila, S. Lacroix, M. Devy & T. Simeon. *Autonomous outdoor mobile robot navigation. The EDEN project*. Lecture Notes in Control and Information Sciences 236. Autonomous Robotic Systems, Eds. A. T. De Almeida, A. Khatib, Springer-Verlag, pages 3–19, Juin 1998.

[Chen 95] M. Chen, T. Jochem & D. Pomerleau. *AURORA : A Vision-Based Roadway Departure Warning System*. In IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS '95), volume 1, pages 243–248, Pittsburgh, Pennsylvannie, August 1995.

[Chen 98] C. H. Chen, L. F. Pau & P. S. P. Wang. The handbook of pattern recognition and computer vision. World Scientific Publishing Co., 1998.

[Cheng 01] H. D. Cheng, X. H. Jiang, Y. Sun & Jingli Wang. *Color image segmentation : advances and prospects*. Pattern Recognition, vol. 34, no. 12, pages 2259–2281, 2001.

[Ciurea 04] Florian Ciurea & Brian Funt. *Tuning Retinex parameters*. Journal of Electronic Imaging, vol. 13, no. 1, pages 58–64, January 2004.

[Cok 87] David R. Cok. Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal, February 10 1987. United States Patent 4,642,678.

[Comaniciu 02] Dorian Comaniciu & Peter Meer. *Mean Shift : A Robust Approach Toward Feature Space Analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pages 603–619, May 2002.

[Comon 94] Pierre Comon. *Independent component analysis, a new concept ?* Signal Processing, vol. 36, no. 3, pages 287–314, April 1994.

[Crane 99] Hewitt D. Crane, John D. Peters & Eugenio Martinez-Uriegas. Method and apparatus for decoding spatiochromatically multiplexed color images using predetermined coefficients, May 4 1999. United States Patent.

[Cross 83] G. R. Cross & A. K. Jain. *Markov Random Field Texture Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 5, no. 1, pages 25–39, January 1983.

[de Albuquerque 04] M. Portes de Albuquerque, I. A. Esquef, A. R. Gesualdi Mello & M. Portes de Albuquerque. *Image thresholding using Tsallis entropy*. Pattern Recognition Letters, vol. 25, no. 9, pages 1059–1065, July 2004.

[Deng 99] Y. Deng, C. Kenney, M. S. Moore & B.S. Manjunath. *Peer group filtering and perceptual color image quantization*. In IEEE Intl. Symposium on Circuits and Systems (ISCAS), volume 4, pages 21–24, Orlando FL, USA, 30 May - 2 June 1999.

[Deng 01] Yining Deng & B. S. Manjunath. *Unsupervised Segmentation of Color-Texture Regions in Images and Video*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 8, pages 800–810, August 2001.

[Derin 87] H. Derin & H. Elliott. *Modeling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 1, pages 39–55, January 1987.

[Dickmanns 88] Ernst Dieter Dickmanns & Volker Graefe. *a) Dynamic monocular machine vision, b) Application of dynamic monocular machine vision.* International Journal of Machine Vision and Applications, vol. 1, no. 4, pages 223–261, November 1988.

[Dickmanns 99] Ernst Dieter Dickmanns. *Computer Vision and Highway Automation.* Vehicle System Dynamics, vol. 31, no. 5, pages 325–343, June 1999.

[Dickmanns 02] Ernst Dieter Dickmanns. *Vision for ground vehicles : history ans prospects.* International Journal of Vehicle Autonomous Systems (IJVAS), vol. 1, no. 1, pages 1–44, 2002.

[Draper 03] Bruce A. Draper, Kyungim Baek, Marian Stewart Bartlett & J. Ross Beveridge. *Recognizing faces with PCA and ICA.* Computer Vision and Image Understanding, vol. 91, no. 1-2, pages 115–137, July 2003.

[Dubins 57] L.E. Dubins. *On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents.* American Journal of Mathematics, vol. 79, pages 497–516, 1957.

[Duda 98] O. Duda, P. E. Hart & David G. Stork. Pattern classification and scene analysis. Wiley & sons, 1998.

[Fairchild 96] M. D. Fairchild. *Refinement of the RLAB color space.* Color Research and Application, vol. 21, no. 5, pages 338–346, 1996.

[Fairchild 98] Mark D. Fairchild. Color appearance models. Addison Wesley, 1998.

[Fernandez-Maloigne 93] C. Fernandez-Maloigne, D. Laugier & C. Boscolo. *Detection of apples with texture analyse for an apple picker robot.* In Intelligent Vehicles '93 Symposium, pages 323–328, July 1993.

[Fernandez-Maloigne 95] C. Fernandez-Maloigne. *Texture and neural network for road segmentation.* In Intelligent Vehicles '95 Symposium, pages 344–349, September 1995.

[Finlayson 95] Graham David Finlayson. *Coefficient Color Constancy.* Thèse de doctorat, Simon Fraser University, School of Computing Science, Burnaby (British Columbia, Canada), 1995.

[Finlayson 00] Graham D. Finlayson & Sabine Süsstrunk. *Performance of a Chromatic Adaptation Transform based on Spectral Sharpening.* In Color Imaging Conference, pages 49–55, Scottsdale, Arizona, July 2000.

[Flusser 92] J. Flusser & T. Suk. *Pattern recognition by affine moments invariants.* Pattern Recognition, vol. 26, no. 1, pages 167–174, January 1992.

[Freeman 88] William T. Freeman. Median filter for reconstructing missing color samples, February 9 1988. United States Patent 4,724,395.

[Frome 04] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow & Jitendra Malik. *Recognizing Objects in Range Data Using Regional Point Descriptors.* In 8th European Conference on Computer Vision, volume 3 of *Lecture Notes in Computer Science*, pages 224–237, Prague, Czech Republic, May 11-14 2004.

[Funt 98] B.V. Funt, K. Barnard & L Martin. *Is colour constancy good enough ?* In Proceedings of the 5th European Conference on Computer Vision, volume 1, pages 445–459, Freiburg, Germany, June 1998.

[Funt 03] Brian V. Funt & Hao Jiang. *Non-von-Kries 3-Parameter Color Prediction*. Proceedings of SPIE, Human Vision and Electronic Imaging VIII, vol. 5007, pages 182–189, June 2003.

[Funt 04] Brian Funt & Florian Ciurea. *Retinex in MATLAB^{TM}*. Journal of Electronic Imaging, vol. 13, no. 1, pages 48–57, January 2004.

[Gasparini 03] F. Gasparini, R. Schettini & P.Gallina. *Tunable cast remover for digital photographs*. In Proceedings of SPIE, volume 5008 of *Color Imaging VIII : Processing, Hardcopy, and Applications*, pages 92–100, January 2003.

[Gasparini 04] F. Gasparini & R. Schettini. *Color balancing of digital photos using simple image statistics*. Pattern Recognition, vol. 37, no. 6, pages 1201–1217, June 2004.

[Geman 84] S. Geman & D. Geman. *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, no. 6, pages 721–741, November 1984.

[Geusebroek 01] Jan-Mark Geusebroek, Rein Van den Boomgaard, Arnold W. M. Smeulders & Hugo Geerts. *Color Invariance*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 12, pages 1338–1349, December 2001.

[Gevers 99] Theo Gevers & Arnold W. M. Smeulders. *Color Based Object Recognition*. Pattern Recognition, vol. 32, no. 3, pages 453–464, March 1999.

[Gevers 02] Theo Gevers. *Adaptive Image Segmentation by Combining Photometric Invariant Region and Edge Information*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 6, pages 848–852, June 2002.

[Ghallab 97] M. Ghallab. *Planification et perception en robotique : Problèmes et approches*. In Journées internationales d' Orsay sur les sciences cognitives (JIOSC'97), La perception du naturel à l'artificiel, pages 5–13, Orsay (Paris), 1-2 décembre 1997.

[Giralt 79] G. Giralt, R. Sobek & R. Chatila. *A Multi-Level Planning and Navigation System for a Mobile Robot; A First Approach to Hilare*. In 6^{th} International Joint Conference on Artificial Intelligence, volume 1, pages 335–337, 1979.

[Gonzalez-Barbosa 04] José-Joël Gonzalez-Barbosa. *Vision Panoramique pour la robotique mobile : stéréovision et localisation par indexation d'images*. Thèse de doctorat, Institut National Polytechnique de Toulouse, LAAS/CNRS, Toulouse (France), Janvier 2004.

[Gonzalez 02] J. J. Gonzalez & S. Lacroix. *Rover Localization in Natural Environments by Indexing Panoramic Images*. In IEEE International Conference on Robotics and Automation (ICRA), volume 2, pages 1365–1370, Washington D.C., USA, May 11-15 2002.

[Guivant 04] J. Guivant, E. Nebot, J. Nieto & F. Masson. *Navigation and Mapping in Large Unstructured Environments*. The International Journal of Robotics Research, vol. 23, no. 4, pages 449–472, April 2004.

[Gunturk 02] Bahadir K. Gunturk, Yucel Altunbasak & Russel Mersereau. *Color Plane Interpolation Using Alternating Projections*. IEEE Image Processing, vol. 11, no. 9, pages 997–1013, September 2002.

[Gunturk 04] Bahadir K. Gunturk, John Glotzbach, Yucel Altunbasak, Ronald W. Schafer & Russell M. Mersereau. *Demosaicking : Color Filter Array Interpolation in Single-Chip Digital Cameras*. IEEE Signal Processing Magazine (Special Issue on Color Image Processing), September 2004.

[Haddad 98] Hassan Al Haddad. *Contrôle par vision du mouvement d'un robot mobile en environnement naturel*. Thèse de doctorat, Université Paul Sabatier (Automatique et Informatique Industrielle), LAAS/CNRS, Toulouse, France, 5 Novembre 1998.

[Hafner 95] James L. Hafner, Harpreet S. Sawhney, William Equitz, Myron Flickner & Wayne Niblack. *Efficient Color Histogram Indexing for Quadratic Form Distance Functions*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 7, pages 729–736, July 1995.

[Hamilton Jr. 97] John F. Hamilton Jr. & James E. Adams Jr. Adaptive color plan interpolation in single sensor color electronic camera, May 13 1997. United States Patent 5,629,734.

[Haralick 79] R. M. Haralick. *Statistical and Structural Approaches to Texture*. Proceedings of the IEEE, vol. 67, no. 5, pages 786–804, May 1979.

[Haralick 93] Robert M. Haralick & Linda G. Shapiro. Computer and robot vision. Addison Weasly, Boston, MA. USA, 1993.

[Hasler 04] David Hasler & Sabine Süsstrunk. *Mapping colour in image stitching applications*. Journal Visual Communication and Image Representation, vol. 15, no. 1, pages 65–90, March 2004.

[Hauta-Kasari 96] M. Hauta-Kasari, J. Parkkinen, T. Jaaskelainen & R. Lenz. *Generalized Co-occurrence Matrix for Multispectral Texture Analysis*. In the 13th International Conference on Pattern Recognition, volume 2, pages 785–789, Vienna, Austria, August 25-30 1996.

[Hayet 02] Jean Bernard Hayet, Frédéric Lerasle & Michel Devy. *A visual landmark framework for indoor mobile robot navigation*. In IEEE International Conference on Robotics and Automation, volume 4, pages 3942–3947. Washington (USA), May 11-15 2002.

[Hayet 03] Jean Bernard Hayet. *Contribution à la navigation d'un robot mobile sur amers visuels texturés dans un environnement structuré*. Thèse de doctorat, Université Paul Sabatier, LAAS/CNRS, Toulouse (France), Janvier 2003.

[Healey 92] Geith Healey. *Segmenting images using normalized color*. EEE Transactions on Systems, Man and Cybernetics, vol. 22, no. 1, pages 64–73, January/February 1992.

[Hu 01] Shi-Min Hu. *Conversion between triangular and rectangular Bézier patches*. Computer Aided Geometric Design, vol. 18, no. 7, pages 667–671, September 2001.

[Huang 90] Chu-Yi Huang, Yen-Shen Chen, Youn-Long Lin & Yu-Chin Hsu. *Data Path Allocation Based on Bipartite Weighted Matching*. In 27^{th} ACM/IEEE Design Automation Conference (DAC), 1990, pages 499–504, Orlando, Florida (USA), June 1990.

[Hutchinson 96] S. A. Hutchinson, G. D. Hager & P. I. Corke. *A tutorial on visual servo control*. IEEE Trans. Robotics and Automation, vol. 12, no. 5, pages 651–670, October 1996.

[Hyvärinen 99] A. Hyvärinen. *Fast and Robust Fixed-Point Algorithms for Independent Component Analysis*. IEEE Transactions on Neural Networks, vol. 10, no. 3, pages 626–634, May 1999.

[Iyatomi 02] Hitoshi Iyatomi & Masafumi Hagiwara. *Scenery image recognition and interpretation using fuzzy inference neural networks*. Pattern Recognition, vol. 35, no. 8, pages 1793–1806, August 2002.

[Jacobson 00] Ralph E. Jacobson, Sidney F. Ray, Geoffrey G. Attridge & Norman R. Axford. Manual of photography, photographic and digital imaging. Focal Press, 9^{th} Edition, 2000.

[Jain 97] A. Jain, N. Ratha & S. Lakshmanan. *Object Detection Using Gabor Filters*. Pattern Recognition, vol. 30, no. 2, pages 295–309, 1997.

[Jochem 95a] T. Jochem, D. Pomerleau, B. Kumar & J. Armstrong. *PANS : A Portable Navigation Platform*. In IEEE Symposium on Intelligent vehicle, pages 107–112, Detroit, Michigan, September 25–26 1995.

[Jochem 95b] T. Jochem, D. Pomerleau & C. Thorpe. *Vision-Based Neural Network Road and Intersection Detection and Traversal*. In IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS '95), volume 3, pages 344–349, Pittsburgh, Pennsylvannie, August 1995.

[Jonker 87] R. Jonker & A. Volgenant. *A Shortest Augmenting Path Algorithm for Dense ans Sparse Linear Assignement Problems*. Computing, vol. 38, no. 4, pages 325–340, March 1987.

[Kass 88] M. Kass, A. Wirkin & D. Terzopoulous. *Snakes : Active Contours Models*. International Journal of Computer Vision, vol. 1, pages 321–331, 1988.

[Katajamäki 03] Juha Katajamäki. *Methods for gamma invariant colour image processing*. Image and Vision Computing, vol. 21, no. 6, pages 527–542, June 2003.

[Katoh 01] Naoya Katoh & Kiyotaka Nakabayashi. *Applying Mixed Adaptation to Various Chromatic Adaptation Transformation (CAT) Models*. In PICS 2001 : Image Processing, Image Quality, Image Capture Systems Conference, volume 4, pages 299–305, Montreal, Quebec, Canada, 2001.

[Kehtarnavaz 02] N. Kehtarnavaz, H.-J. Oh & Y. Yoo. *Development and real-time implementation of auto white balancing scoring algorithm*. Real-Time Imaging, vol. 8, no. 5, pages 379–386, October 2002.

[Kelly 98] Alonzo Kelly & Anthony Stentz. *Rough Terrain Autonomous Mobility - Part 1 : A Theoretical Analysis of Requirements and Part 2 : An Active Vision, Predictive Control*. Autonomous Robots, vol. 5, no. 2, pages 129–198, May 1998.

[Khadraoui 98] D. Khadraoui, R. Rouveure, C. Debain, P. Martinet, P. Bonton & J. Gallice. *Vision Based Control in Driving Assistance of Agricultural Vehicles*. International Journal of Robotics Research, vol. 17, no. 10, pages 1040–1054, October 1998.

[Kim 96] Dong-Woo Kim, Gee-Hyuk Lee & Soo-Yong Kim. *Stochastic Segmentation of Severely Degraded Images Using Gibbs Random Fields*. OPTICAL REVIEW, vol. 3, no. 3, pages 184–191, May/June 1996.

[Kimmel 99] Ron Kimmel. *Demosaicing : Image Reconstruction from Color CCD Samples*. IEEE Transactions on Image Processing, vol. 8, no. 9, pages 1221–1228, September 1999.

[Kittler 86] J. Kittler & J. Illingworth. *Minimum Error Thresholding*. Pattern Recognition, vol. 19, no. 1, pages 41–47, January 1986.

[Kosaka 92] A. Kosaka & A. C. Kak. *Fast Vision-Guided Mobile Robot Navigation using Model-Based Reasoning and Prediction of Uncertainties*. Computer Vision Graphics, and Image Processing - Image Understanding, vol. 56, no. 3, pages 271–329, November 1992.

[Kraft 99] J. M. Kraft & D. H. Brainard. *Mechanisms of color constancy under nearly natural viewing*. Proceedings of the National Academy of Sciences USA., vol. 96, pages 307–312, January 1999.

[Kries 93] J. Von Kries. *Chromatic Adaptation*. In Festschrift der Albrecht-Ludwigs-Universität, 1902 [Translation : D. L. MacAdam, "Colorimetry-Fundamentals"], SPIE, volume MS 77 of *Milestone Series*, 1993.

[Krumm 95] J. Krumm & S. Shafer. *Texture Segmentation and Shape in the Same Image*. In The Fifth International Conference on Computer Vision, volume 1, pages 121–127, Cambridge, Massachusetts, June 20 - 23 1995.

[Kuipers 91] B. Kuipers & Y. T. Byum. *A robot exploration and mapping strategy based on semantic hierarchy representations*. Journal of Robotics and Autonomous Systems, vol. 8, pages 47–63, 1991.

[Kumar 03] Sanjiv Kumar, Alexander C. Loui & Martial Hebert. *An Observation-Constrained Generative Approach for Probabilistic Classification of Image Regions*. Image and Vision Computing, vol. 21, no. 1, pages 87–97, January 2003.

[Lacroix 02] S. Lacroix, I-K. Jung & A. Mallet. *Digital Elevation Map Building with Low Altitude Stereo Imagery*. Robotics and Autonomous Systems, vol. 41, no. 2-3, pages 119–127, November 30 2002.

[Lamiraux 01] F. Lamiraux & J.P. Laumond. *Smooth Motion Planning for car-like vehicules*, august 2001.

[Land 71] E. H. Land & J. J. McCann. *Lightness and retinex theory*. JOSA, vol. 61, no. 1, pages 1–11, 1971.

[Laroche 94] Claude A. Laroche & Mark A. Prescott. Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients, December 13 1994. United States Patent 5,373,322.

[Lasserre 96] Patricia Lasserre. *Vision pour la robotique mobile en environnement naturel*. Thèse de doctorat, Université Paul Sabatier, LAAS/CNRS, Toulouse, France, 26 Septembre 1996.

[Lauzière 99] Yves Bérubé Lauzière, Denis Gingras & Frank P. Ferrie. *Color camera characterization with an application to detection under daylight*. In Proceedings of the Vision Interface'99 Conference, pages 280–287, Trois-Rivieres, Québec, Canada, 18–21, May 1999.

[Lee 94] J. H. Lee, B. H. Chang & S. D. Kim. *Comparison of colour transformations for image segmentation*. Electronics Letters, vol. 30, no. 20, pages 1660–1661, September 1994.

[Lee 99] W. S. Lee, D. C. Slaughter & D. K. Giles. *Robotics Weed Control System for Tomatoes*. Precision Agriculture, vol. 1, pages 95–113, 1999.

[Lemonde 04] V. Lemonde & M. Devy. *Obstacle detection with stereovision*. In Mechatronics & Robotics (MECHROB'04), volume 3, pages 919–924, Aachen (Allemagne), September 13-15 2004.

[Levitt 87] T. S. Levitt. *Qualitative navigation*. In editor Morgan Kaufmann, editeur, Image Understanding Workshop, 1987.

[Li 97] Liyuan Li, Jian Gong & Weinan Chen. *Grey-Level Image Thresholding Based on Fisher Linear Projection of two-Dimensional Histogram*. Pattern Recognition, vol. 30, no. 5, pages 743–749, May 1997.

[Li 03] Shutao Li, James T. Kwok, Hailong Zhu & Yaonan Wang. *Texture classification using the support vector machines*. Pattern recognition, vol. 36, no. 12, pages 2883–2893, December 2003.

[Longère 02] Philippe Longère, Xuemei Zhang, Peter B. Delahunt & David H. Brainard. *Perceptual assessment of demosaicing algorithm performance*. Proceedings of the IEEE, vol. 90, no. 1, pages 123–132, January 2002.

[Lorigo 97] L. M. Lorigo, R. A. Brooks & W. E. L. Grimson. *Visually-Guided Obstacle Avoidance in Unstructured Environments*. In IEEE/RSJ International Conference on Intelligence Robots and Systems, volume 1, pages 373–379, Grenoble, France, September 1997.

[Lucassen 97] Marcel P. Lucassen & Jan Walraven. *Color Constancy under Natural and Artificial Illumination.* Vision Research, vol. 36, no. 17, pages 2699–2711, September 1997.

[Luo 97] Jiebo Luo, Robert T. Gray & Hsien-Che Lee. *Towards physics-based segmentation of photographic color images.* In IEEE International Conference on Image Processing (ICIP'97), volume 3, pages 58–61, Washington, DC, October 26–29 1997.

[Malik 90] J. Malik & P. Perona. *Preattentive Texture Discrimination with Early Vision Mechanisms.* Journal of Optical Society America A., vol. 7, no. 5, pages 923–932, May 1990.

[Malis 99] Ezio Malis, François Chaumette & Sylvie Boudet. *2-1/2-D Visual Servoing.* IEEE Transactions on Robotics and Automation, vol. 15, no. 2, pages 238–250, April 1999.

[Mallet 00] A. Mallet, S. Lacroix & L. Gallo. *Position estimation in outdoor environments using pixel tracking and stereovision.* In IEEE International Conference on Robotics and Automation, volume 4, pages 3519–3524, San Francisco (USA), April 2000.

[Mancuso 01] Massimo Mancuso & Sebastiano Battiato. *An Introduction to the Digital Still Camera Technology.* ST Journal of System Research - Special Issue on Image Processing for Digital Still Camera, vol. 2, no. 2, pages 1–9, December 2001.

[Marin-Hernandez 99] A. Marin-Hernandez & H. V. Rios-Figueroa. *Eels : Electric Snakes.* Computación y Sistemas, vol. 2, no. 2-3, pages 87–94, 1999.

[Marin-Hernandez 04] Antonio Marin-Hernandez. *Vision dynamique pour la navigation d'un robot mobile.* Thèse de doctorat, Institut National Polytechnique, LAAS, Toulouse, France, 9 Février 2004.

[Matas 95] J. Matas, M. Marik & J. Kittler. *On representation and matching of multi-coloured objects.* In International Conference on Computer Vision, pages 726–732, 1995.

[Mateus 05] D. Mateus, J. G. Avina-Cervantes & M. Devy. *Robot visual navigation in semi-structured outdoor environments.* In IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain, April 18-22 2005.

[Meng 93] M. Meng & A. C. Kak. *Mobile robot navigation using neural networks and nonmetrical environmental models.* IEEE Control Systems Magazine, vol. 13, no. 5, pages 30–39, October 1993.

[Merrill 03] Dick Merrill. *The Next-Generation Digital Camera.* Optics and Photonics News, pages 26–34, January 2003.

[Miura 97] Jun Miura & Yoshiaki Shirai. *Vision and Motion Planning for a Mobile Robot under Uncertainty.* International Journal of Robotics Research, vol. 16, no. 6, pages 806–825, December 1997.

[Miura 02] Jun Miura, Motokuni Itoh & Yoshiaki Shirai. *Towards Vision-Based Intelligent Navigator : Its Concept and Prototype.* IEEE Transactions on Intelligent Transportation Systems, vol. 3, no. 2, pages 136–146, June 2002.

[Muresan 02] D. Darian Muresan & Thomas W. Parks. *Optimal Recovery Demosaicing.* In The 4th IASTED International Conference on Signal and Image Processing, Hawaii, USA, August 12-14 2002.

[Murrieta-Cid 98] Rafael Murrieta-Cid. *Contribution au développement d'un système de Vision pour robot mobile d'extérieur.* Thèse de doctorat, Institut National Polytechnique de Toulouse, LAAS/CNRS, Toulouse, France, Novembre 1998.

[Murrieta-Cid 01] R. Murrieta-Cid, C. Parra, M. Devy & M. Briot. *Scene modeling from 2D and 3D sensory data acquired from natural environments.* In IEEE The 10th International Conference on Advanced Robotics, pages 221–228, Budapest, Hungary, August 22-25 2001.

[Murrieta-Cid 02] R. Murrieta-Cid, C. Parra, M. Devy, B. Tovar & C. Esteves. *Building Multi-Level Models : From Landscapes to Landmarks*. In IEEE International Conference on Robotics and Automation, volume 4, pages 4346–4353, Washington, D.C., 2002.

[Novak 90] C. L. Novak, S. A. Shafer & R. G. Wilson. *Obtaining Accurate Color Images for Machine Vision Research*. In Perceiving, Measuring and Using Color, SPIE Proceedings, volume 1250, pages 54–68, February 1990.

[Ohlander 78] R. Ohlander, K. Price & D Raj Reddy. *Picture Segmentation using a Recursive Region Splitting Method*. Computer Graphics and Image Processing, vol. 8, pages 313–333, 1978.

[Ohno 97] Yoshihiro Ohno & Jonathan E. Hardis. *Four-Color Matrix Method for Correction of Tristimulus Colorimeters*. In IS&T, Fifth Color Imaging Conference, pages 301–305, Scottsdale, Arizona, USA, November 16-20 1997.

[Ohta 80] Y. Ohta, T. Kanade & T. Sakai. *Color Information for region segmentation*. Computer Graphics and Image Processing, vol. 13, no. 3, pages 222–241, July 1980.

[Ohta 85] Y. Ohta. Knowledge-based interpretation of outdoor natural color scenes. Research Notes in Artificial Intelligence 4, Pitman Advanced Publishing Program, 1985.

[Ojala 96] T. Ojala, M. Pietikäinen & D. Harwood. *A comparative study of texture measures with classification based on feature distributions*. Pattern Recognition, vol. 29, no. 1, pages 51–59, 1996.

[Orteu 89] J. J. Orteu. *Segmentation d'image en régions par la méthode de Séparation-Fusion (Split and Merge)*. Rapport de recherche 89127, LAAS/CNRS, Avril 1989.

[Otsu 79] N. Otsu. *A Threshold Selection Method from Gray-Level Histograms*. I.E.E.E. Transactions on Systems, Man and Cybernetics, vol. 9, no. 1, pages 62–66, January 1979.

[Pal 91] Nikhil R. Pal & Sankar K. Pal. *Image Model, Poisson Distribution and Object Extraction*. International Journal of Pattern Recognition and Artificial Intelligence, vol. 5, no. 3, pages 459–483, 1991.

[Pal 93] Nikhil R. Pal & Sankar K. Pal. *A review on image segmentation techniques*. Pattern Recognition, vol. 26, no. 9, pages 1277–1294, September 1993.

[Paletta 00] L. Paletta, M. Prantl & A. Pinz. *Learning Temporal Context in Active Object Recognition Using Bayesian Analysis*. In 15th International Conference on Pattern Recognition (ICPR 2000), volume 1, pages 695–699, Barcelona, Spain, September 3-8 2000.

[Pan 98] J. Pan, G. N. Desouza & A. C. Kak. *Fuzzy Shell : a large-scale expert system shell using fuzzy logic for uncertainty reasoning*. IEEE Transactions on Fuzzy Systems, vol. 6, no. 4, pages 563–581, November 1998.

[Parra-Rodriguez 99] Carlos Alberto Parra-Rodriguez. *Contribution à la modélisation topologique par vision 2D et 3D pour la navigation d'un robot mobile sur terrain naturel*. Thèse de doctorat, Université Paul Sabatier, LAAS/CNRS, Toulouse (France), Mars 1999.

[Pettre 03] Julien Pettre. *Planification de mouvements de marche pour acteurs digitaux*. Thèse de doctorat, Université Paul Sabatier, LAAS, Toulouse, France, Novembre 2003.

[Plataniotis 00] K. N. Plataniotis & A. N. Venetsanopoulos. Color image processing and applications. Springer-Verlag, 2000.

[Pomerleau 96] D. Pomerleau & T. Jochem. *Rapidly Adapting Machine Vision for Automated Vehicle Steering*. IEEE Expert : Special Issue on Intelligent System and their Applications, vol. 11, no. 2, pages 19–27, April 1996.

[Pratt 96] Ian Pratt. *Shape Representation using Fourier Coefficients of the Sinusoidal Transform*. Technical report umcs-96-7-1, University of Manchester, 1996.

[Qi 01] Yuan Qi, David Doermann & Daniel DeMenthon. *Hybrid Independent Component Analysis and Support Vector Machine Learning Scheme for Face Detection*. In IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, volume 3, pages 1481–1484, Salt Lake City, Utah, May 2001.

[Ramanath 02] Rajeev Ramanath, Wesley E. Snyder, Griff L. Bilbro & William A. Sander III. *Demosaicking methods for Bayer color arrays*. Journal of Electronic Imaging, vol. 11, no. 3, pages 306–315, July 2002.

[Ramanath 03] A. Rajeev Ramanath & B. W. Snyder. *Adaptive demosaicking*. Journal of Electronic Imaging, vol. 12, no. 4, pages 633–642, October 2003.

[Rasmussen 02] Christopher Rasmussen. *Combining Laser Range, Color and Texture Cues for Autonomous Road Following*. In IEEE International Conference on Robotics and Automation, volume 4, pages 4320–4325, Washington, DC, USA, May 11-15 2002.

[Rasmussen 04a] Christopher Rasmussen. *Grouping dominant orientations for ill-structured road following*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 470–477, Washington, DC, June 27 - July 2 2004.

[Rasmussen 04b] Christopher Rasmussen. *Texture-Based Vanishing Point Voting for Road Shape Estimation*. In British Machine Vision Conference (BMVC), Kingston Univeristy, London, September 7-9 2004.

[Reeds 90] J. A. Reeds & R. A. Shepp. *Optimal paths for a car that goes both forward and backwards*. Pacific Journal of Mathematics, vol. 145, no. 2, pages 367–393, 1990.

[Regensburger 94] U. Regensburger & V. Graefe. *Visual recognition of obstacles on roads*. In IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, IROS'94, volume 2, pages 980–987, Munic, Germany, September 12-16 1994.

[Reinhard 01] Erik Reinhard, Michael Ashikhmin, Bruce Gooch & Peter Shirley. *Color Transfer between Images*. IEEE Computer Graphics and Applications, vol. 21, no. 5, pages 34–41, September/October 2001.

[Reyna-Rojas 02] R. A. Reyna-Rojas. *Conception et intégration VLSI d'un système de vision générique. Application à la détection et à la localisation d'objets à l'aide de « Support Vector Machines »*. Thèse de doctorat, LAAS - CNRS, Toulouse, France, Avril 2002.

[Rives 97] P. Rives & J. Borrelly. *Underwater Pipe Inspection Task Using Visual Servoing Techniques Techniques*. In IEEE/RSJ International Conference on Intelligence Robots and Systems, volume 1, pages 63–68, Grenoble, France, September 1997.

[Rizzi 98] A. Rizzi, G. Bianco & R. Cassinis. *A Bee-Inspired Visual Homing Using Color Images*. Robotics and Autonomous Systems, vol. 25, no. 3-4, pages 159–164, November 1998.

[Rizzi 03] Alessandro Rizzi, Carlo Gatta & Daniele Marini. *A new algorithm for unsupervised global and local color correction*. Pattern Recognition Letters, vol. 24, no. 11, pages 1663–1677, July 2003.

[Rosenberg 01] Charles Rosenberg, Martial Hebert & Sebastian Thrun. *Image Color Constancy Using KL-Divergence*. In Eighth IEEE International Conference on Computer Vision, volume 1, pages 239–246, Vancouver, Canada, July 7-14 2001.

[Rubner 98] Y. Rubner, C. Tomasi & L. J. Guibas. *A Metric for Distributions with Applications to Image Databases*. In the 1998 IEEE International Conference on Computer Vision, pages 59–66, Bombay,(India), January 1998.

[Saber 96] E. Saber, A. M. Tekalp, R. Eschbach & K. Knox. *Automatic Image Annotation Using Adaptative Color Classification.* Graphical Models and Image Processing, vol. 58, no. 2, pages 115–126, March 1996.

[Sahoo 04] Prasanna K. Sahoo & Gurdial Arora. *A thresholding method based on two-dimensional Renyi's entropy.* Pattern Recognition, vol. 37, no. 6, pages 1149–1161, June 2004.

[Sakamoto 98] T. Sakamoto, C. Nakanishi & T. Hase. *Software pixel interpolation for digital still cameras suitable for a 32-bit MCU.* IEEE Transactions on Consumer Electronics, vol. 44, no. 4, pages 1342–1352, November 1998.

[Sánchez-Yáñez 03a] Raúl E. Sánchez-Yáñez & Antonio Fernández Evguenii V. Kurmyshev. *One-class texture classifier in the CCR feature space.* Pattern Recognition Letters, vol. 24, no. 9-10, pages 1503–1511, June 2003.

[Sánchez-Yáñez 03b] Raúl E. Sánchez-Yáñez, Evguenii V. Kurmyshev & Francisco J. Cuevas. *A framework for texture classification using the coordinated clusters representation.* Pattern Recognition Letters, vol. 24, no. 1-3, pages 21–31, January 2003.

[Serrano 04] Navid Serrano, Andreas E. Savakis & Jiebo Luo. *Improved scene classification using efficient low-level features and semantic cues.* Pattern Recognition, vol. 37, no. 9, pages 1773–1784, September 2004.

[Sharma 97] Gaurav Sharma & H. Joel Trussell. *Digital Color Imaging.* IEEE Transactions on Image Processing, vol. 6, no. 7, pages 901–932, July 1997.

[Siebert 01] Andreas Siebert. *Retrieval of gamma corrected images.* Pattern Recognition Letters, vol. 22, no. 2, pages 249–256, February 2001.

[Siegwart 04] Roland Siegwart & Illah R. Nourbakhsh. Introduction to autonomous mobile robots. The MIT Press, 2004.

[Simhon 98] Saul Simhon & Gregory Dudek. *A global topological map formed by local metric maps.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), volume 3, pages 1708–1714, Victoria, Canada, October 1998.

[Stewart 93] G. W. Stewart. *On the Early History of the Singular Value Decomposition.* SIAM Review, vol. 35, no. 4, pages 551–566, December 1993.

[Süsstrunk 01] Sabine Süsstrunk, Jack Holm & Graham D. Finlayson. *Chromatic Adaptation Performance of Different RGB Sensors.* In IS&T/SPIE Electronic Imaging, SPIE, volume 4300, pages 172–183, January 2001.

[Swain-Oropeza 99] Ricardo Swain-Oropeza. *Contrôle de tâches référencées vision pour la navigation d'un robot mobile en milieu structuré.* Thèse de doctorat, Institut National Polytechnique de Toulouse, LAAS/CNRS, Toulouse (France), Juin 1999.

[Swain 91] Michael J. Swain & Dana H. Ballard. *Color Indexing.* International Journal Computer Vision, vol. 7, no. 1, pages 11–32, November 1991.

[Talukder 02] A. Talukder, R. Manduchi, A.Rankin & L. Matthies. *Fast and Reliable Obstacle Detection and Segmentation for Cross Country Navigation.* In IEEE Intelligent Vehicles Symposium 2002. Versailles (France), June 18-20 2002.

[Thorpe 88] C. Thorpe, M. H. Hebert, T. Kanade & S. A. Shafer. *Vision and Navigation for the Carnegie-Mellon Navlab.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 3, pages 362–373, May 1988.

[Thorpe 03] Charles Thorpe, Justin David Carlson, David Duggins, Jay Gowdy, Robert MacLachlan, Christoph Mertz, Arne Suppe & Chieh-Chih Wang. *Safe Robot Driving in Cluttered Environments*. In 11th International Symposium of Robotics Research, Siena, Italy, October 19-22 2003.

[Thrun 98] S. Thrun. *Learning Metric-Topological Map for Indoor Mobile Robot Navigation*. Artificial Intelligence, vol. 99, no. 1, pages 21–71, February 1998.

[Torralba 01] Antonio B. Torralba & Pawan Sinha. *Statistical context priming for object detection*. In The Eighth IEEE International Conference on Computer Vision, (ICCV-01), volume 1, pages 763–770. Vancouver, British Columbia, Canada, July 9–12 2001.

[Torralba 03] Antonio B. Torralba. *Contextual Priming for Object Detection*. International Journal of Computer Vision, vol. 53, no. 2, pages 169–191, 2003.

[Trémeau 04] Alain Trémeau, Christine Fernandez-Maloigne & Pierre Bonton. Image numérique couleur. DUNOD, 2004.

[Tsin 01] Yanghai Tsin, Visvanathan Ramesh & Takeo Kanade. *Statistical Calibration of the CCD Imaging Process*. In IEEE International Conference on Computer Vision (ICCV), volume 1, pages 480–487, Vancouver, Canada, July 9-12 2001.

[Tsugawa 94] Sadayuki Tsugawa. *Vision-Based Vehicles in Japan : Machine Vision Systems and Driving Control Systems*. IEEE Transactions on Industrial Electronics, vol. 41, no. 4, pages 398–405, August 1994.

[Tsumura 86] T. Tsumura. *Survey of automated guided vehicle in a Japanese factory*. In IEEE International Conference on Robotics and Automation, volume 3, pages 1329–1334, April 1986.

[Turk 88] M. A. Turk, D. G. Morgenthaler, K. D. Gremban & M. Marra. *VITS-A Vision System for Autonomous Land Vehicle Navigation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, no. 3, pages 342–361, May 1988.

[Unser 86] M. Unser. *Sum and difference histograms for texture classification*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 1, pages 118–125, January 1986.

[Vapnik 96] V. N. Vapnik, S. E. Golowich & A. J. Smola. *Support vector method for function approximation, regression estimation, and signal processing*. Advances in Neural Information Processing Systems (NIPS), vol. 9, pages 281–287, December 2-5 1996.

[Vapnik 98] Vladimir N. Vapnik. Statistical learning theory. John Wiley & Sons, Inc., 1998.

[Veltkamp 99] R. C. Veltkamp & M. Hagedoorn. *State of Art in Shape Matching*. Technical report uu-cs-1999-27, Utrecht University, Netherlands, 1999.

[Veltkamp 01] Remco C. Veltkamp. *Shape Matching : Similarity Measures and Algorithms*. Technical report uu-cs-2001-03, Utrecht University, Netherlands, January 25 2001.

[Wang 90] Li Wang & Dong-Chen He. *exture classification using Texture Spectrum*. Pattern Recognition, vol. 23, no. 8, pages 905–910, 1990.

[Wang 03] Kongqiao Wang & Jari A. Kangas. *Character location in scene images from digital camera*. Pattern Recognition, vol. 36, no. 10, pages 2287–2299, October 2003.

[Ward 02] Greg Ward & Elena Eydelberg-Vileshin. *Picture Perfect RGB Rendering Using Spectral Prefiltering and Sharp Color Primaries*. In 13^th Eurographics Workshop Rendering, Pisa, Italy, June 26-28 2002.

[Wilcox 92] B. Wilcox, L. Matthies, D. Gennery, B. Cooper, T. Nguyen, T. Litwin, A. Mishkin & H. Stone. *Robotic vehicles for planetary exploration.* In IEEE International Conference on Robotics and Automation, volume 1, pages 175–180, May 12-14 1992.

[Yan 03] Chengxin Yan, Nong Sang & Tianxu Zhang. *Local entropy-based transition region extraction and thresholding.* Pattern Recognition Letters, vol. 24, no. 16, pages 2935–2941, December 2003.

[Yuen 02] Pong Chi Yuen & Jian Huang Lai. *Face representation using independent component analysis.* Pattern Recognition, vol. 35, no. 6, pages 1247–1257, June 2002.

[Zabih 94] R. Zabih & J. Woodfill. *Non-parametric Local Transforms for Computing Visual Correspondence, (ECCV).* In Third European Conference on Computer Vision, volume 2, pages 151–158, Stockholm, Sweden, May 1994.

[Zhang 04] Dengsheng Zhang & Guojun Lu. *Review of shape representation and description techniques.* Pattern Recognition, vol. 37, no. 1, pages 1–19, January 2004.

[Zhong 00] Yu Zhong & Anil K. Jain. *Object localization using color, texture and shape.* Pattern Recognition, vol. 33, no. 4, pages 671–684, April 2000.

[Zoller 02] Thomas Zoller, Lothar Hermes & Joachim M. Buhmann. *Combined color and texture segmentation by parametric distributional clustering.* In 16th International Conference on Pattern Recognition, volume 2, pages 627–630, 2002.

[Zomet 02] Assaf Zomet & Shmuel Peleg. *Multi-sensor super-resolution.* In Sixth IEEE Workshop on Applications of Computer Vision, WACV, pages 27–31, Orlando, Florida, December 3-4 2002.

Résumé

Cette thèse porte sur le traitement automatique d'images couleur, et son application à la robotique dans des environnements semi-structurés d'extérieur. Nous proposons une méthode de navigation visuelle pour des robots mobiles en utilisant une caméra couleur. Les domaines d'application de ce travail se situent dans l'automatisation de machines agricoles, en vue de la navigation automatique dans un réseau de chemins (pour aller d'une ferme à un champ par exemple).

Nous présentons tout d'abord une analyse des principaux travaux de recherche dans la littérature sur la navigation visuelle. Une chaîne de pré-traitement pour le rendu couleur d'images numériques mono-capteur dotées d'un filtre Bayer est présentée ; elle se base sur une étude des techniques de démosaïquage, le calibrage chromatique d'images (balance de blancs) et la correction gamma.

Une méthode d'interprétation monoculaire de la scène courante permet d'extraire les régions navigables et un modèle 2D de la scène. Nous traitons de la segmentation d'une image couleur en régions, puis de la caractérisation de ces régions par des attributs de texture et de couleur, et enfin, de l'identification des diverses entités de la scène courante (chemin, herbe, arbre, ciel, champ labouré,...). Pour cela, nous exploitons deux méthodes de classification supervisée : la méthode de *Support Vector Machine* (SVM) et celle des k plus proches voisins (k-PPV). Une réduction d'information redondante par une analyse en composantes indépendantes (ACI) a permis d'améliorer le taux global de reconnaissance.

Dans un réseau de chemins, le robot doit reconnaître les intersections de chemins lui permettant (a) dans une phase d'apprentissage, de construire un modèle topologique du réseau dans lequel il va devoir se déplacer et (b) dans une phase de navigation, de planifier et exécuter une trajectoire topologique définie dans ce réseau. Nous proposons donc une méthode de détection et classification du chemin : ligne droite, virage gauche, virage droite, carrefour en X, en T ou en Y. Une approche pour la représentation de la forme et de la catégorisation des contours (*Shape Context*) est utilisée à cet effet. Une validation a été effectuée sur une base d'images de routes ou chemins de campagne. En exploitant cette méthode pour détecter et classifier les noeuds du réseau de chemins, un modèle topologique sous forme d'un graphe est construit ; la méthode est validée sur une séquence d'images de synthèse.

Enfin, dans la dernière partie de la thèse, nous décrivons des résultats expérimentaux obtenus sur le démonstrateur Dala du groupe Robotique et IA du LAAS-CNRS. Le déplacement du robot est contrôlé et guidé par l'information fournie par le système de vision à travers des primitives de déplacement élémentaires (Suivi-Chemin, Suivi-Objet, Suivi-Bordure,...). Le robot se place au milieu du chemin en construisant une trajectoire à partir du contour de cette région navigable. Étant donné que le modèle sémantique de la scène est produit à basse fréquence (de 0,5 à 1 Hz) par le module de vision couleur, nous avons intégré avec celui-ci, un module de suivi temporel des bords du chemin (par *Snakes*), pour augmenter la fréquence d'envoi des consignes (de 5 à 10 Hz) au module de locomotion. Modules de vision couleur et de suivi temporel doivent être synchronisés de sorte que le suivi puisse être réinitialisé en cas de dérive. Après chaque détection du chemin, une trajectoire sur le sol est planifiée et exécutée ; les anciennes consignes qui ne sont pas encore exécutées sont fusionnées et filtrées avec les nouvelles, donnant de la stabilité au système.

Mots-clés: segmentation couleur, texture, classification, topologie, robotique mobile, navigation visuelle

Abstract

This thesis deals with the automatic processing of color images, and its application to robotics in outdoor semi-structured environments. We propose a visual-based navigation method for mobile robots by using an on-board color camera. The objective is the robotization of agricultural machines, in order to navigate automatically on a network of roads (to go from a farm to a given field).

Firstly, we present an analysis of the main research works about visual-based navigation literature. A preprocessing chain for color rendering on mono-sensor digital images equipped with a Bayer filter, is presented ; it is based on the analysis of the demosaicking techniques, the chromatic calibration of images (white point balance) and the correction gamma.

Our monocular scene interpretation method makes possible to extract the navigable regions and a basic 2D scene modeling. We propose functions for the segmentation of the color images, then for the characterization of the extracted regions by texture and color attributes, and at last, for their classification in order to recognize the road and other entities of the current scene (grass, trees, clouds, hedges, fields,...). Thus, we use two supervised classification methods : Support Vector Machines (SVM) and k nearest neighbors (k-NN). A redundancy reduction by using independent components analysis (ICA) was performed in order to improve the overall recognition rate.

In a road network, the robot needs to recognize the roads intersections in order to navigate and to build a topological model from its trajectory. An approach for the road classification is proposed to recognize : straight ahead, turn-left, turn-right, road intersections and road bifurcations. An approach based on the road shape representation and categorization (*shape context*) is used for this purpose. A validation was carried out on an image dataset of roads or country lanes. By exploiting this method to detect and classify the nodes of a road network, a topological model based on a graph is built ; the method is validated on a sequence of synthetic images.

Finally, Robot displacement is controlled and guided by the information provided by the vision system through elementary displacement primitives (Road-Follow, Follow-Object, Follow-Border,...). Robot Dala is placed in the middle of the road by computing a trajectory obtained from the navigable region contours. As retrieving semantic information from vision is computationally demanding (low frequency $0.5 \approx 1$ Hz), a *Snakes* tracking process was implemented to speed up the transfer of instructions ($5 \approx 10$ Hz) to the locomotion module. Both tasks must be synchronized, so the tracking can be re-initialized if a failure is detected. Locomotion tasks are planned and carried out while waiting for new data from the vision module ; the instructions which are not yet carried out, are merged and filtered with the new ones, which provides stability to the system.

Keywords: color segmentation, texture, classification, topology, mobile robotics, visual navigation