Localisation et détection de fermeture de boucle basées saillance visuelle : algorithmes et architectures matérielles

Merwan Birem

N° d'ordre : 2558

EDSPIC : 691

Université Blaise Pascal - Clermont II

École Doctorale
Sciences Pour L'ingénieur de Clermont-Ferrand

Thèse

Présentée par

Merwan BIREM

Pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPÉCIALITÉ : Électronique et architecture des systèmes

Titre de la thèse :

Localisation et détection de fermeture de boucle basées sur la saillance visuelle : algorithmes et architectures matérielles

soutenue publiquement le **12 Mars 2015** devant le jury :

Directeur de thèse :	M. François Berry	Maître de Conférences
Co Directeur de thèse :	M. Youcef Mezouar	Professeur des Universités
	M. Jean-Charles Quinton	Maître de Conférences
Rapporteur :	M. David Filliat	Professeur des Universités
	M. Lionel Lacassagne	Maître de Conférences
Président du jury :	M. Fabrice Meriaudeau	Professeur des Universités

RÉSUMÉ

Dans plusieurs tâches de la robotique, la vision est considérée comme l'élément essentiel avec lequel la perception de l'environnement ou l'interaction avec d'autres utilisateurs peut se réaliser. Néanmoins, les artefacts potentiellement présents dans les images capturées rendent la tâche de reconnaissance et d'interprétation de l'information visuelle extrêmement compliquée. Il est de ce fait, très important d'utiliser des primitives robustes, stables et ayant un taux de répétabilité élevé afin d'obtenir de bonnes performances.

Cette thèse porte sur les problèmes de localisation et de détection de fermeture de boucle d'un robot mobile en utilisant la saillance visuelle. Les résultats en termes de précision et d'efficacité des applications de localisation et de détection de fermeture sont évalués et comparés aux résultats obtenus avec des approches de l'état de l'art sur différentes séquences d'images acquises en milieu extérieur. Le principal inconvénient avec les modèles proposés pour l'extraction de zones de saillance est leur complexité de calcul, ce qui conduit à des temps de traitement important. Afin d'obtenir un traitement en temps réel, nous présentons dans ce mémoire l'implémentation du détecteur de régions saillantes sur la plate forme reconfigurable *DreamCam*.

MOTS-CLÉS

Attention Visuelle, Caméra intelligente, Détection de fermeture de boucle, FPGA, Localisation de robot, Primitives visuelles, Systèmes embarqués, Temps réel, Traitement d'images.

ABSTRACT

In several tasks of robotics, vision is considered to be the essential element by which the perception of the environment or the interaction with other users can be realized. However, the potential artifacts in the captured images make the task of recognition and interpretation of the visual information extremely complicated. It is therefore very important to use robust, stable and high repeatability rate primitives to achieve good performance.

This thesis deals with the problems of localization and loop closure detection for a mobile robot using visual saliency. The results in terms of accuracy and efficiency of localization and closure detection applications are evaluated and compared to the results obtained with the approaches provided in literature, both applied on different sequences of images acquired in outdoor environnement. The main drawback with the models proposed for the extraction of salient regions is their computational complexity, which leads to significant processing time. To obtain a real-time processing, we present in this thesis also the implementation of the salient region detector on the reconfigurable platform *DreamCam*.

KEYWORDS

Embedded systems, FPGA, Image processing, Robot localization, Loop closure detection, Real time, Smart camera, Visual attention, Visual features.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMS

ABTM Appearance Based Topological Mapping

ACP Analyse en Composantes Principales

ALM high performance Adaptation Logic Modules

ARM Advanced Reduced Instruction Set Computer (RISC) Machines

ASIC Application Specific Integrated Circuit

BRIEF Binary Robust Independent Elementary Features

BRISK Binary Robust Invariant Scalable Key-points

BoW Bag of Words

CBIR Content Based Image Retrieval

CC City Center

CCD Charge Coupled Device

CE Chip Enable

CIE Commission Internationale de l'Éclairage

CLB Configurable Logic Block

CMOS Complementary Metal Oxide Semiconductor

CML Concurrent Mapping and Localization

CPU Central Processing Unit

CZ CEZEAUX

CUDA Compute Unified Device Architecture

DLCF Département Laboratoire de Clermont-Ferrand

DSP Digital Signal Processing

ECC Error Checking and Correcting

EMD Earth Mover Distance

EVT Eye Vision Technology

FABMAP Fast Appearance Based Mapping

FAST Features from Accelerated Segment Test

FIFO First In First Out

FIT Feature Integration Theory

FOA Focus of Attention

FPGA Field Programmable Gate Array

GbE Gigabit Ethernet

GIST « The central idea - the essence »

GMM Generalized Method of Moments

GPU Graphics Processing Unit

GPS Global Position System

HD Haute Définition

HDL Hardware Description Language

HPL High Performance Low power

HSV Hue Saturation Value

I2C Inter Integrated Circuit

IBR Intensity Based Regions

IOB Input Output Block

IOR Inhibition Of Return

IP Institut Pascal

IPDS Institut Pascal Data Set

ISP Image Sequence Partitioning

JTAG Joint Test Action Group

LASMEA Laboratoire des Sciences et Matériaux pour l'Électronique, et d'Automatique

LE Logic Element

LIDAR LIght Detection And Ranging

LUT Look-Up Table

MSER Maximally Stable Extensible Regions

MSR Most Salient Region

NC New College

NVT Neuromorphic Vision Toolkit

NPI Nombre de Point d'Intérêt

ORB Orientation Binary Robust Independent Elementary Features (BRIEF)

OE Output Enable

PC Personal Computer

PCI Peripheral Component Interconnect

PV PAVIN

PLL Phase Locked Loop

RAM Random Access Memory

RISC Reduced Instruction Set Computer

RS Region Saillante

RTL Register Transfer Logic

ROI Region Of Interest

ROM Read Only Memory

RVB Rouge, Vert et Bleu

SHARC Streaming Hardware Accelerator with Run-time Configurability

SIFT Scale Invariant Feature Transform

SIMD Single Instruction Multiple Data

SLAM Simultaneous Localization And Mapping

SoC System on Chip

SRAM Static Random Access Memory

SSD Sum of Squared Difference

SURF Speeded Up Robust Features

SUSAN Smallest Univalue Segment Assimilating Nucleus

TSMC Taiwan Semiconductor Manufacturing Company

USB Universal Serial Bus

VOCUS Visual Object detection with a CompUtational attention System

VHDL Very High Speed Integrated Circuit (VHSIC) Hardware Description Language

VHSIC Very High Speed Integrated Circuit

VGA Video Graphics Array

VLSI Very-large-scale integration

WTA Winner-Takes-All

WE Write Enable

ZNCC Zero-mean Normalized Cross Correlation

Première partie

INTRODUCTION

"I have not failed 700 times. I have not failed once. I have succeeded in proving that those 700 ways will not work. When I have eliminated the ways that will not work, I will find the way that will work."

"Je n'ai pas échoué 700 fois. Je n'ai pas échoué une seule fois. J'ai réussi à prouver que ces 700 façons ne marcheront pas. Lorsque j'aurai éliminé les façons qui ne fonctionnent pas, j'aurai trouvé la façon qui fonctionnera"

Thomas Edison, about creating the light bulb.

INTRODUCTION GÉNÉRALE

INTRODUCTION

Regarder est une activité tellement naturelle pour les êtres humains et les animaux que l'ampleur de la tâche est souvent sous-estimée lorsqu'on essaie de donner la vue à une machine. L'invention de la caméra et de l'ordinateur ont permis de franchir un premier pas vers la vision artificielle, mais le plus dur reste à faire. La caméra est capable d'acquérir une image, un peu à la manière de ce que fait l'œil. Elle peut même dépasser les performances de l'œil dans certaines situations. L'avantage des caméras est qu'elles sont des capteurs peu onéreux fournissant une information très riche de l'environnement. Mais pour interpréter l'image, l'ordinateur et les algorithmes mis au point par l'homme sont loin d'égaler les capacités du cerveau.

Pour faire face à l'énorme quantité d'informations visuelles de notre environnement visuel, le système visuel possède la faculté de sélectionner une information pertinente localisée spatialement dans le champ visuel parmi toutes celles qui lui parviennent : on parle d'attention visuelle. L'attention visuelle se réfère à la capacité des systèmes de vision à sélectionner rapidement les données les plus pertinentes, c'est-à-dire les plus saillantes dans une scène. L'objectif principal de ce comportement visuel est de réduire considérablement la quantité d'informations visuelles qui doivent être traitées par des tâches plus complexes de haut niveau, telles que la reconnaissance d'objet.

Dans le cadre de la robotique mobile, la vision est considérée comme l'élément essentiel avec lequel la perception de l'environnement ou l'interaction avec d'autres utilisateurs peut se réaliser. De fait, le robot mobile doit accomplir deux tâches fondamentales : cartographier son environnement et se localiser. Pour réaliser ces deux tâches, le robot doit être capable d'extraire de l'information de l'environnement.

A l'heure actuelle, la plupart des architectures de traitement d'images mises en œuvre pour l'extraction d'information, sont souvent basées sur des ordinateurs. Ainsi, les données visuelles sont acquises par une ou plusieurs caméras et les images sont directement transmises vers le Personal Computer (PC) hôte. Cette transmission de gros volumes de données est un problème récurrent bien connue dans les systèmes de vision. Une solution permettant de répondre à ce problème, et qui constitue le cadre applicatif de cette thèse, est l'utilisation d'un système de vision embarqué de type caméra intelligente ou smart camera. L'un des objectifs des caméras intelligentes est d'offrir la possibilité d'extraire des primitives (flux optique, détection de coin, régions saillantes, ...) de la scène observée directement au sein du système d'acquisition. Une fois ces primitives calculées, elles sont ensuite envoyées vers le PC hôte réduisant ainsi notablement le volume de données transmis.

Cette thèse, s'inscrivant dans le domaine de la perception visuelle sur système embarqué pour la robotique, est donc connexe aux domaines de la robotique mobile, de la vision par ordinateur et des systèmes embarqués. Elle traite, dans le contexte du Simul-

taneous Localization And Mapping (SLAM) topologique, du problème de la détection visuelle de fermeture de boucle. Cet algorithme est crucial à la fois pour la localisation du robot et pour la cartographie. Dans le contexte de la vision et des systèmes embarqués, elle traite du problème de l'implémentation d'algorithmes complexes de traitement d'images sur caméra intelligente.

OBJECTIFS

Les travaux effectués lors de cette thèse peuvent être regroupés en deux axes principaux :

- Un axe algorithmique dans lequel nous présentons des algorithmes de détection de primitives utilisés en robotique. Les applications dans lesquelles, de telles primitives sont utilisées comme par exemple : la détection de fermeture de boucle et la construction de carte topologique. Nous nous intéressons plus particulièrement aux primitives de type régions saillantes et leurs utilisations dans des applications de robotique mobile.

- Un axe architecture regroupant la présentation d'une nouvelle caméra intelligente nommée *DreamCam*. En plus de cela, on présente dans cette partie les implémentations matérielles des algorithmes de détection de primitives présentés dans la première partie.

Les objectifs des recherches visés par cette thèse sont au nombre de quatre :

1. Étude des techniques de navigation de robot mobile. Comparaison et classification de ces techniques en vue du développement d'une nouvelle approche pour la détection de fermeture de boucle.

2. Étude des algorithmes d'extraction de primitives de type régions saillantes et points d'intérêt, et leurs utilisations dans le domaine de la robotique. La figure ci-dessous donne un aperçu des deux composantes principales d'un extracteur de primitives à savoir : le détecteur et le descripteur.

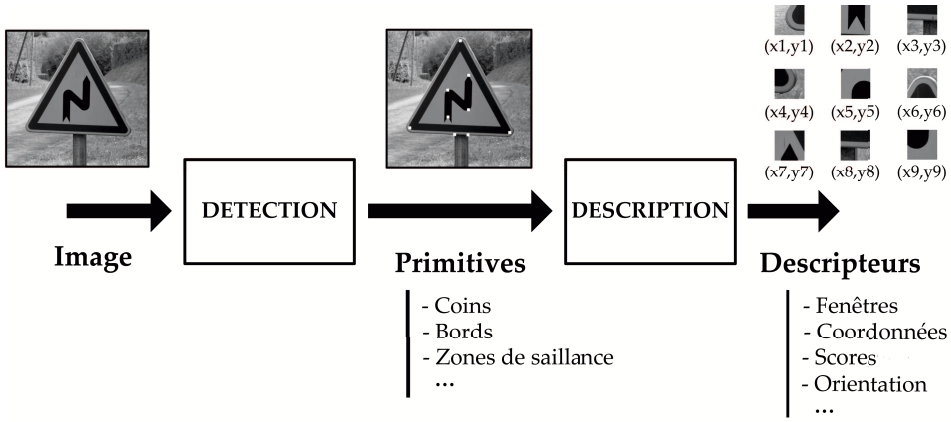


FIGURE 0.1: Extraction de primitives.

3. Réalisation d'un système de navigation. Dans lequel deux processus sont traités : le premier de détection de fermeture de boucle, et le second de création de carte topologique. L'utilisation de régions saillantes pour les deux processus avec une approche traditionnelle et une approche probabiliste suivant un modèle bayesien.

4. Implémentation de deux algorithmes d'extraction de primitive sur caméra intelligente. Le premier algorithme est celui de Harris & Stephen détectant des points d'intérêt. Cet algorithme est souvent utilisé par les roboticiens pour des applications de mise en correspondance et reconstruction 3D. Le second algorithme détecte des régions saillantes suivant un modèle bio-inspiré. Ce second type de primitives étant choisi pour les différents avantages qu'elles offrent : facilement détectable, nombre réduit sur une image et taux de répétabilité élevé. L'implémentation d'un tel algorithme sur un capteur intelligent permet à la fois d'isoler les informations importantes et nécessaires au fonctionnement des algorithmes de haut niveau, mais aussi de diminuer grandement la quantité de données à transmettre.

CONTRIBUTIONS

Les travaux présentés dans ce manuscrit ont permis la rédactions de cinq articles scientifiques. Quatre d'entre eux ont déjà été publiés dans des conférences et journaux internationaux.

- Dans les deux articles [124, 125], nous présentons les travaux correspondant à la première partie du manuscrit. Dans le premier article [124] les primitives de type régions saillantes sont utilisées pour la détection de fermeture de boucle. Ces primitives sont d'abord détectées avec le modèle Frintrop (2006) [56], ensuite décrites en utilisant le descripteur crées. Ce descripteur rend plus robustes les régions saillantes vis-à-vis des changements qui peuvent arrivés dans la scène observée (présence ou absence d'objet, changement de point de vue et d'éclairage de la scène). Les différents résultats présentés dans cet article ont été comparés à l'algorithme de détection de fermeture de boucle Fast Appearance Based Mapping (FABMAP). Dans le second article [125] les régions saillantes sont toujours utilisées avec les même détecteur et descripteur, mais avec une approche probabiliste pour la détection de fermeture de boucle, la localisation et la construction de la mémoire visuelle représentant l'environnement.

- Dans les articles [123, 19], nous présentons les travaux correspondant à la seconde partie du manuscrit. Dans l'article [123] nous donnons l'architecture matérielle de la caméra intelligente *DreamCam*. et l'implémentation matérielle d'algorithme d'extraction de points d'intérêt, c'est-à-dire le détecteur ainsi que le descripteur. L'article [19] présente uniquement l'implémentation de la partie détection de points d'intérêt (l'algorithme de Harris & Stephen).

- Dans un autre article en cours de rédaction, nous présentons l'implémentation que l'on propose pour la détection de régions saillantes sur la *DreamCam*.

STRUCTURE DU MANUSCRIT

CHAPITRE 1 : Dans ce chapitre nous introduisons quelques notions en robotique mobile, essentielles à la comprehension du manuscrit. Les primitives locales et globales sont présentées dans ce chapitre pour montrer leur importance et leur utilisation pour des applications en : détection de fermeture de boucle, reconstruction 3D, mise en correspondance et cartographie.

CHAPITRE 2 : Ce chapitre présente un bref état de l'art sur les différents modèles bio-inspirés pour la détection des régions saillantes en commençant par le modèle pionnier de Koch and Ullman (1987) [98]. Le modèle de Frintrop (2006) [56] utilisé et étendu dans nos implémentations est présenté d'une façon plus détaillée.

CHAPITRE 3 : Dans ce chapitre nous présentons les deux techniques que nous avons développées basées sur les régions saillantes, pour la détection de fermeture de boucle et pour la localisation.

CHAPITRE 4 : Ce chapitre présente la chaîne de traitement d'images standard, la classification des algorithmes sous trois niveaux d'abstraction et les différents architectures embarquées de traitements numériques des images en temps réel.

CHAPITRE 5 : Nous présentons dans ce chapitre l'architecture de la *DreamCam*, une caméra intelligente à base d'un Field Programmable Gate Array (FPGA). Afin de montrer les avantages d'un tel système, nous présentons également l'algorithme Harris & Stephen et son implémentation sur FPGA. Cet algorithme est utilisé pour la détection de points d'intérêt.

CHAPITRE 6 : Dans ce chapitre, nous présentons l'implémentation de l'algorithme de détection de régions saillantes présenté au chapitre 2. Les différents résultats concernant la consommation de ressources FPGA et fréquence maximale sont exposés.

CONCLUSION : Enfin nous faisons un bilan sur les différentes contributions et nous présentons nos perspectives de développement et de recherches futures.

Deuxième partie

ALGORITHMIQUE

1

NAVIGATION DE ROBOT : ÉTAT DE L'ART

1.1 INTRODUCTION

Ces dernières années, les systèmes robotiques autonomes ont connu un développement considérable dans divers domaines applicatifs, allant de la réalisation de tâches simples pour les robots ménagers, à la réalisation de taches plus compliquées et dangereuses comme par exemple le désamorçage de bombe. La Figure 1.1 montre quelques exemples de robots existants.

FIGURE 1.1: Exemple de robots [52].

Les robots mobiles sont apparus dans les années 70, peu après les robots manipulateurs ; ceux-ci peuvent être à roues, aériens, sous-marins ou marcheurs. Jusqu'aux années 2000, les recherches ont été focalisées essentiellement sur les robots à roues, simples d'un point de vue mécanique, et se déplaçant dans un environnement contrôlé. De nos jours, grâce aux progrès considérables dans différents domaines tels que l'informatique, la mécatronique et fort de l'expérience acquise sur les robots mobiles à roues,

les recherches portant sur l'utilisation d'autres types de robots, dans différentes conditions et dans des environnements peu ou pas structurés[1] sont en plein développement.

Par définition un robot mobile est une machine automatique qui est capable de faire un mouvement particulier dans un environnement qui peut être contrôlé ou non. Suivant le type de mouvement on parlera alors de navigation globale, lorsque le robot se déplace d'un point « A » à un point « B » dans son environnement, ou locale lorsque le robot interagit avec des objets/éléments de son environnement (saisie d'objet par exemple)[99].

La navigation globale de robot mobile peut être classée en trois catégories : Dans la première, le robot ne nécessite pas de carte représentant l'environnement et la navigation est réalisée de manière réactive. Dans la deuxième catégorie, le robot dispose de la carte de l'environnement et pour naviguer en toute sécurité, le robot doit constamment estimer et corriger, si nécessaire, sa position dans la carte. Dans la troisième catégorie, le robot est dans un environnement complètement ou partiellement inconnu, la construction de la carte (*cartographie*) est faite à la volée lors de la navigation du robot dans l'environnement.

Nous introduisons dans la suite de ce chapitre certaines notions indispensables à la compréhension des objectifs de ce mémoire. Celles-ci concernent le domaine de la robotique et la vision par ordinateur.

1.2 CARTOGRAPHIE

Parmi les fonctionnalités essentielles requises par les robots mobiles autonomes, on s'intéresse ici en particulier à la cartographie. Celle-ci représente la capacité de construire une carte représentant l'environnement à corriger ou mettre à jour une carte préconstruite. Une des motivations à la cartographie réside dans le fait que les robots ne disposent pas toujours de cartes ou celles-ci peuvent être incomplètes ou erronées.

1.2.1 *Concepts*

L'aspect clé de la cartographie est de percevoir l'environnement et de représenter l'information perçue comme une carte utilisable par le robot. Cette information peut être extraite à l'aide de capteurs disponibles sur le marché, allant de capteurs infrarouges bas coût à des radars très onéreux. Suivant le type de capteur utilisé, l'information perçue sera différente. Par exemple, une caméra va projeter le monde tridimensionnel sur une image bidimensionnelle, tandis qu'un télémètre laser va fournir des informations sur la distance qu'il y a entre le capteur et les objets qui l'entourent. Les informations fournies par ces différents capteurs sont souvent bruitées, ce qui entrave le processus de construction de la carte, entraînant des incohérences dans la carte finale.

1 Environnement structuré : terminologie consacrée pour dire que la géométrie de l'environnement dans lequel évolue le robot est connu.[15]

1.2.2 *Types de cartes*

Les cartes construites peuvent être classées suivant leurs représentations en trois catégories :

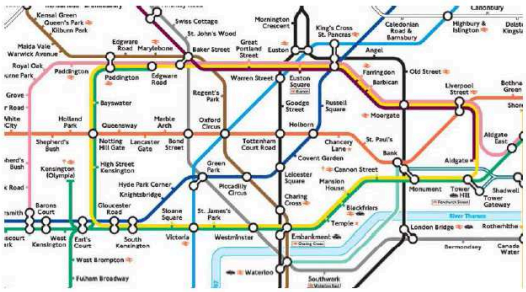
MÉTRIQUE : Une carte métrique décrit la position du robot et la position de tous les objets détectés dans l'environnement dans un référentiel géométrique absolu. Les cartes géographiques de la Figure 1.2a sont des exemples courants de cartes métriques. La Figure 1.2b montre une carte métrique construite par un robot en utilisant les données acquises à l'aide d'un télémètre laser.

(a) Un exemple classique de cartes métriques.

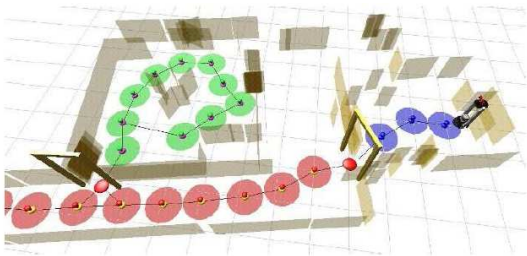
(b) Un exemple de carte métrique construite par un robot.

FIGURE 1.2: Exemple de cartes métriques [99].

TOPOLOGIQUE : Contrairement aux cartes métriques, une carte topologique ne décrit pas la position du robot ou des objets dans un référentiel géométrique absolu mais représente l'environnement sous la forme d'un graphe[2]. Un exemple classique de carte topologique est la carte de métro (Figure 1.3a) dans laquelle les nœuds représentent les différentes stations et les arêtes reliant des paires de nœuds indiquent la traversabilité entre eux. La Figure 1.3b illustre un exemple de carte topologique créée dans un environnement intérieur par un robot mobile.



(a) Un exemple classique de cartes topologiques.



(b) Un exemple de carte topologique construite par un robot.

FIGURE 1.3: Exemple de cartes topologiques [99].

HYBRIDE : Il est également possible d'utiliser des représentations hybrides qui vont avoir des caractéristiques à la fois topologiques et métriques afin de bénéficier des avantages de chacune des deux représentations précédentes.

2 Un graphe $G = (N, A)$ est un ensemble fini non nul de nœuds $N = \{N_1, N_2, \ldots, N_n\}$ associé à un ensemble d'arcs (ou arêtes) $A = \{a_1, a_2, \ldots, a_n\}$ où chaque arc peut être défini par deux sommets $a_r = a_{s,t} = (N_s, N_t)$. G est un *graphe orienté* ou *digraphe* si tous les éléments de A sont des arêtes orientées ($a_{s,t} \neq a_{t,s}$).

1.2.3 *Cartographie basée capteurs*

Une cartographie de l'environnement peut être faite de différentes manières et avec différents niveaux de détail ; les propriétés de la carte dépendent principalement de la tâche à réaliser par le robot, du type d'environnement, ainsi que de la résolution et du champ de vue des capteurs montés sur le robot. Parmi les modalités sensorielles qui peuvent être employées pour cartographier un environnement il y a la télémétrie (lasers, sonars, radar), la vision (monoculaire, stéréoscopique, et panoramique), le toucher (utilisation de pare-chocs tactiles). Les mesures issues de ces capteurs sont souvent combinées à des données proprioceptives (odométriques ou inertielles), goniométriques (boussoles) ou à des données numériques Global Position System (GPS).

Parmi l'ensemble des capteurs possibles dont un robot mobile peut disposer, ce sont les capteurs de vision, les caméras, qui fournissent au robot le plus d'information sur l'environnement. Ces capteurs sont bon marché et peu gourmands en énergie [74].

La vision est souvent choisie comme capteur privilégié du fait de la richesse de l'information perçue : résolution, information aussi bien géométrique que radiométrique (luminance, couleur). La richesse de l'information visuelle rend possible une interprétation contextuelle du contenu des images acquises dans un environnement particulier ; par exemple dans le contexte de la conduite automobile les feux tricolores et les panneaux de signalisation sont parmi les informations recherchées dans les images [74].

1.3 CARTOGRAPHIE TOPOLOGIQUE

Dans le cas d'environnements de grande taille, la construction d'une représentation purement métrique nécessite la mémorisation de grande quantité de données, ce qui peut s'avérer lourd à manipuler.

Une alternative à cette approche, est l'approche de cartographie topologique, où l'environnement est représenté par un graphe, dans lequel chaque nœud correspond à un endroit caractéristique, appelé *lieu* par la suite. Chaque lieu devra être décrit par un ensemble de caractéristiques propres, qui permettront au robot de le reconnaître. Ce que les nœuds et les arêtes représentent dans une carte topologique dépend de l'application, du type de capteur et des algorithmes utilisés pour la construire. Dans le cas de la navigation autonome, une arête liant deux nœuds signifie qu'il existe une ou plusieurs commande(s) que le robot peut exécuter afin de se déplacer entre les deux lieux correspondants [74]. Ainsi, la navigation entre deux lieux associés à deux nœuds non adjacents dans le graphe, empruntera un chemin déterminé par une séquence de commandes permettant de se déplacer entre les nœuds intermédiaires [32]. Ce concept fonctionne sur l'hypothèse que les lieux (ou nœuds) 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 nœud au suivant dans la carte.

En dépit de sa relative fréquence, il n'existe pas de consensus précis sur ce qu'est une carte topologique ou comment elle devrait être construite. Les théories de la cartographie cognitive qui tentent d'expliquer comment les êtres humains cartographient les espaces à grande échelle, suppose que les cartes cognitives ont une forte nature topologique [35, 69, 102].

1.3.1 *Cartographie topologique basée vision*

Les approches de cartographie topologique basées sur la vision sont communément appelées : « Appearance Based Topological Mapping (ABTM) ». Ces approches sont de plus en plus populaires pour les raisons suivantes :

- Les images fournissent des informations riches de l'environnement avec la présence de couleurs et de textures.
- La représentation des données sous forme d'images compactes, par opposition aux énormes nuages de points fournis par les capteurs télémètres laser ou les « LIght Detection And Ranging (LIDAR)s ».
- La compréhension sémantique de l'environnement est relativement plus facile avec des données visuelles que des données télémétriques [110].
- Les caméras utilisées pour la navigation ayant des performances bonnes sont largement disponibles et bon marché.

1.3.2 *Densité des cartes*

Dans les approches de cartographie basée vision, certaines considèrent chaque image acquise en tant que nœud dans la carte topologique. Ces techniques peuvent être considérées comme étant des approches de cartographie topologique *dense* car il y a autant de nœuds dans la carte que d'observations. Il existe d'autres techniques pour représenter la carte avec moins de nœuds et cela en partitionnant la séquence d'images (Image Sequence Partitioning (ISP)) de sorte que toutes les images ayant la même apparence soient représentées par un seul nœud dans la carte topologique. Comme le nombre de nœuds dans ce processus sera inférieur au nombre total d'images d'entrée, les cartes obtenues sont appelées des cartes topologiques *éparses*. Dans la suite de ce paragraphe, nous nous intéressons uniquement à ce type de cartes.

Des cartes topologiques construites dans des environnements d'intérieur sont présentées dans [204] et [205]. Dans ces travaux la carte topologique de l'environnement est segmentée en utilisant l'algorithme de coupes de graphes normalisé, ce qui va générer des sous cartes correspondant à des zones concaves dans l'environnement. Dans [100] ce sont des primitives visuelles (points-clés = key-points) qui ont été utilisées pour effectuer l'appariement d'images sur une séquence complète. Ils détectent les transitions entre emplacements intérieurs en fonction du nombre de primitives qui peuvent être appariés avec succès entre les images successives.

Les travaux mentionnés ci-dessus ont été expérimentés sur des environnements intérieurs qui contiennent des espaces concaves (comme les salles) et sont relativement faciles à partitionner par rapport aux environnements extérieurs qui, en général, n'ont pas de frontières claires séparant les lieux.

Une technique de classification spectrale[3] est appliquée pour la cartographie topologique dans les travaux de Valgren et al. (2007) [185]. Les images sont groupées à l'aide

3 En intelligence artificielle, le terme classification spectrale désigne une famille d'algorithmes de classification non-supervisée. Cette dernière est de plus en plus utilisée, à la fois en raison de son efficacité, et de sa

des similarités codées dans *la matrice de similarité*[4], puis ces images sont représentées en tant que nœuds dans la carte topologique. Il en résulte un graphe topologique éparse. Une autre technique d'ISP a été présentée dans les travaux de Nourani-Vatani and Pradalier (2010) [144] qui utilise la moyenne de la norme du vecteur de flux optique pour signaler les changements dans la scène et former des nœuds à ces endroits uniquement. Murillo et al. (2010) [135] ont construit une carte topologique éparse en utilisant l'ISP avec des primitives invariantes à la rotation nommées omni-gist [147].

1.4 MÉTHODES DE CARTOGRAPHIE

1.4.1 *Cartographie en-ligne*

Afin de construire *en-ligne* (temps réel) la carte représentant l'environnement, le robot se déplace dans cet environnement tout en se localisant dans la carte en cours de construction, pour pouvoir positionner dans la carte les éléments observés (amers). D'une part, la localisation peut être erronée soit à cause des informations bruitées fournies par le(s) capteur(s), soit à cause de l'aliasing perceptuel (phénomène pour lequel deux endroits significativement différents paraissent similaires). Cette fausse localisation du robot, donne lieu à des cartes erronées. D'autre part, pour se localiser avec précision, il faut que la carte préconstruite soit également précise, ce qui reste difficile. Pour résumer, une construction de carte précise ne peut se faire sans un processus de localisation précis. Le robot doit être capable d'estimer sa position dans l'environnement tout en le cartographiant au fur et à mesure de sa progression. Il faut donc être en mesure d'assurer le bouclage entre localisation et cartographie de l'environnement. Ce problème de « l'œuf et de la poule » est bien connu des roboticiens, qui le désignent sous le nom de localisation et cartographie simultanée ou SLAM en anglais voire plus rarement Concurrent Mapping and Localization (CML). Dans la suite de ce mémoire, c'est le terme de SLAM qui sera utilisé exclusivement. Le SLAM a été mis en œuvre dans de nombreux environnements difficiles en dehors des environnements intérieurs et extérieurs habituels. Quelques scénarios incluent des environnements sous-marins [28], des environnements souterrains dangereux [145] et des applications spatiales [183].

La problématique du SLAM implique en outre la gestion de multiples sous-problèmes [176]. Il faut d'abord gérer les erreurs de perception[5] et de localisation (dérive des capteurs proprioceptifs, imprécision des mesures extéroceptives, présence de bruit, erreurs d'interprétation, ...). A partir de ces mesures sensorielles imparfaites, il faut alors mettre en correspondance les observations locales entachées d'erreur avec les éléments de la carte globale en cours de construction. Dans certains cas, il faut également ajouter le traitement des environnements dynamiques et pour les robots totalement autonomes, des capacités d'exploration. Ainsi, selon certains chercheurs comme Thrun et al. (2002)

simplicité relative d'implémentation qui se résume principalement à l'extraction de vecteurs propres d'une matrice de similarités.

4 La matrice de similarité contient les différents score de similarités entre l'ensemble des pairs d'images de la base de données.

5 La notion de perception en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure. Lorsqu'il s'agit de naviguer de manière autonome dans des lieux totalement ou partiellement inconnus, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'environnement dans lequel il évolue pour extraire les informations utiles à l'accomplissement de sa tâche. Le choix des capteurs dépend bien évidemment de l'application envisagée.

[176], la cartographie automatique peut être considérée comme le problème perceptuel le plus difficile en robotique mobile : les progrès dans ce domaine devraient avoir des retombées sur de nombreux autres aspects de la robotique tels que les interactions homme(s)/robot(s) ou la coordination multi-robot par exemple [49].

L'algorithme le plus fréquemment utilisé pour traiter le problème du SLAM est fondé sur l'utilisation d'un filtre de Kalman étendu [91]. Dans cet algorithme, un vecteur d'état est utilisé contenant à la fois la position courante du robot et celles de l'intégralité des amers[6] précédemment observés dans l'environnement. Ce vecteur est associé à une matrice de covariance qui permet de mesurer la précision de l'état estimé, c'est-à-dire, la position du robot et des amers. Lorsqu'une nouvelle image est acquise la position du robot est mise à jour ainsi que la position des amers observés. En général, l'incertitude sur les amers est réduite à chaque nouvelle observation.

La principale limite de cet algorithme est le temps de calcul de la mise à jour du filtre de Kalman qui augmente avec la taille de la carte. La complexité de ce calcul est en $O(N^2)$, où N est la taille du vecteur d'état, proportionnelle au nombre d'amers observés. Cela veut dire que faire du SLAM temps-réel avec l'algorithme décrit plus haut n'est possible que dans des environnements de taille réduite où le nombre d'amers utilisés est limité. Pour résoudre ce problème, il est possible d'utiliser des sous-cartes. La mise à jour est faite uniquement pour une carte locale de l'environnement immédiat du robot. Leonard and Newman (2003) [108] ont proposé un algorithme pour gérer la cohérence entre les sous-cartes.

1.4.2 *Cartographie hors-ligne*

Afin de palier le problème de la cartographie en-ligne, c'est-à-dire, du SLAM, il est possible de considérer séparément le problème de la cartographie et de la localisation. La partie la plus complexe, qui est la cartographie, est traité *hors-ligne* pour obtenir un système de localisation rapide capable de fournir la position du robot en temps réel par la suite.

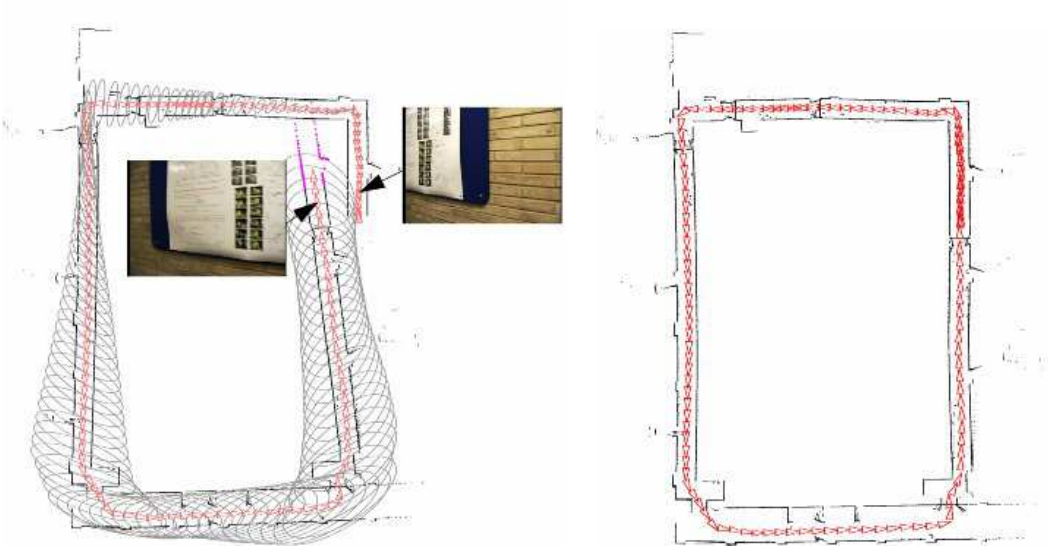
Dans cette approche, comme pour la précédente il est possible de cartographier un environnement en utilisant uniquement la vision ou en la couplant avec un ou plusieurs autres types de capteurs. Par exemple, Cobzas et al. (2003) [37] utilisent en plus de la caméra placée sur une plate forme rotative un télémètre laser, afin d'obtenir au final une carte contenant un ensemble d'images panoramiques enrichies de l'information de profondeur. La localisation est faite sur cette carte à partir d'image 2D seulement. Kidono et al. (2002) [94] utilisent pour la cartographie : une tête stéréoscopique et un odomètre. La localisation du robot est faite ensuite sur la carte 3D en temps réel. Ohya et al. (2001) [146] construisent également une carte 3D en utilisant un capteur trinoculaire et un odomètre. Cette carte contient la position des lignes verticales observées durant la phase d'apprentissage. Li and Tsuji (1999) [111] proposent une approche différente de construction de carte 3D. Dans cette approche, la caméra est placée sur un véhicule de façon à observer le côté de la route (vision latérale). A partir de ces observations, une segmentation fondée sur le mouvement permet de différencier les façades des différents bâtiments qui sont classés selon la distance à la caméra.

6 Un **amer** est un point de repère fixe et identifiable sans ambiguïté utilisé pour la navigation maritime.

Dans les travaux de Kelly (2000) [92] une mosaïque du sol est utilisée comme carte 2D pour guider un véhicule automatisé servant au transport dans un bâtiment industriel. Le même concept a été utilisé par Gracias and Santos-Victor (2000) [63] pour localiser un robot submersible aux abords d'une canalisation sous-marine.

1.5 FERMETURE DE BOUCLE

Les deux approches de cartographie, métriques et topologiques, s'appuient fortement sur la détection de fermeture de boucle, aussi connue sous le nom de reconnaissance de lieu, pour la réussite de la construction de carte. Le processus de détection de fermeture de boucle doit permettre de déterminer si le robot est en train de réexaminer un emplacement précédemment visité ou s'il est dans un tout nouvel emplacement. La Figure 1.4 illustre une situation dans laquelle le robot revisite un emplacement de manière à former une boucle. Ceci est un exemple de fermeture de boucle lors de la construction d'une carte métrique. La figure montre également l'incertitude sur la position du robot tout au long de la trajectoire. Dans le cas des cartes topologiques, la fermeture de boucle capte la structure topologique de l'environnement en mettant à jour la carte ou en créant des arrêtes entre les nœuds adjacents.



(a) L'incertitude en position avant la fermeture de la boucle.

(b) L'incertitude en position après la fermeture de la boucle.

FIGURE 1.4: Exemple de détection de fermeture de boucle [99].

Cependant, le processus de détection de fermeture de boucle est un problème difficile pour les raisons suivantes :

- ÉVOLUTIVITÉ : Comme la taille de la carte augmente au fur et à mesure de sa construction, chaque nouvelle observation doit être comparée avec toutes les observations précédentes déjà enregistrées/représentées dans la carte.

- ALIASING PERCEPTUEL : Le problème de l'aliasing perceptuel désigne l'incapacité d'un système de perception à distinguer de manière unique tous les lieux d'un environnement. Deux endroits différents dans l'environnement mais qui semblent

similaires (aux capteurs employés) peuvent conduire à des faux positifs lors de l'évaluation de la correspondance, ce qui altère la précision du système.

- BRUIT : Chaque mesure est accompagnée d'un certain degré de bruit dépendant du type de capteur utilisé et de l'environnement dans lequel les mesures sont acquises.

- VARIABILITÉ : Il existe dans l'environnement des paramètres incontrôlables et variables tel que l'illumination lorsqu'une caméra est utilisée ce qui complique le problème de mise en correspondance d'images.

- OBJETS DYNAMIQUES : Les objets dynamiques sont un problème dans les deux environnements intérieurs et extérieurs, de ce fait l'évaluation de correspondance doit être robuste en présence de tels objets.

Il est important de comprendre la différence qu'il y a entre la localisation et la fermeture de boucle. Le processus de détection de fermeture de boucle doit classer chaque observation en lieu revisité ou en nouvel endroit. Tandis que, dans le processus de localisation, on suppose que toute nouvelle observation est nécessairement issue d'un endroit qui est dans la carte donnée. Pour cette raison, la fermeture de boucle est considérée comme un problème beaucoup plus complexe que la localisation [66, 171]. La localisation peut être considérée comme un cas particulier de la détection de fermeture de boucle dans lequel l'hypothèse que le robot se trouve forcément dans une partie connue de l'environnement est faite.

1.5.1 *Fermeture de boucle vs Recherche d'images par le contenu*

Le processus de détection de fermeture de boucle a quelques points en commun avec le processus de recherche d'images en fonction de leurs contenus (Content Based Image Retrieval (CBIR)). Le CBIR vise généralement en la recherche d'images, en récupérant les images les plus similaires dans une grande base de données ou en donnant un indice à partir d'une requête. Une requête peut être constituée de mots-clés textuels décrivant l'image (en cas de recherche d'images sur le web), de textures, de couleurs, d'un objet dans l'image ou bien d'images complètes. Le processus de détection de fermeture de boucle ressemble au CBIR lorsque la requête est une image. Les deux processus peuvent se différencier par les points suivants :

- Le CBIR vise à récupérer les images les plus pertinentes dans l'ordre décroissant de pertinence. Tandis que le processus de détection de fermeture de boucle cherche à trouver l'image correspondante la plus précise, un résultat nul est obtenu si l'image correspondante n'est pas précise. Par conséquent, il est important de distinguer un cas de correspondance d'un cas de non-correspondance.

- Dans le CBIR chaque image de la base de données est indépendante de toutes les autres. Ceci n'est pas le cas dans une démarche de cartographie où les images sont stockées dans leur ordre d'acquisition, ce qui garantit la disponibilité des images voisines de toute image de référence donnée. La disponibilité des images voisines peut être utilisée pour imposer des contraintes temporelles supplémentaires ([9, 16]) sur l'appariement, ce qui permet de réduire les fausses correspondances.

- Alors que le CBIR est un processus basé uniquement sur l'image, le processus de détection de fermeture de boucle peut utiliser d'autres informations provenant d'autres types de capteurs, proprioceptifs comme l'odomètre ou extéroceptifs comme les télémètres laser ou les radars.

Faire du CBIR ou de la détection de fermeture de boucle revient à mettre en correspondance des images acquises dans des conditions différentes. En effet, afin de réaliser cette tâche (présentée dans la section 1.6), une représentation sous forme convenable et compacte des images est nécessaire. Cela se fait en sélectionnant l'information contenue dans l'image en gardant les composantes pertinentes, qui constituent les primitives visuelles locales ou globales (voir Figure 1.5).

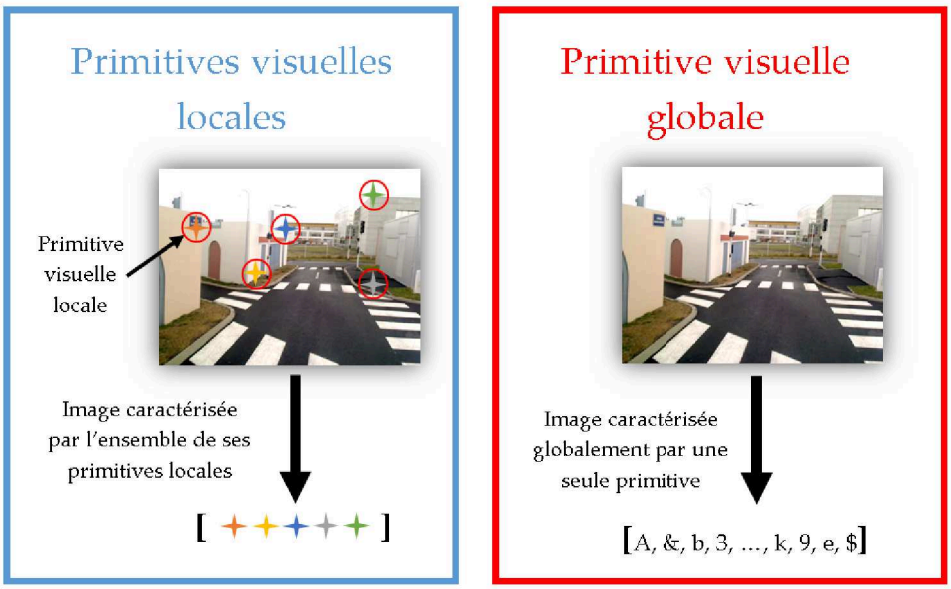


FIGURE 1.5: Primitives visuelles. Une image peut être caractérisée par l'ensemble des primitives locales qu'elle contient ou bien par une seule primitive globale encodant ses caractéristiques générales.

Le processus de détection de fermeture de boucle recherche si l'image courante est déjà indexée dans la carte préconstruite. Si c'est le cas cela signifie que le robot a déjà visité cet endroit. Dans le cas contraire si l'image courante ne correspond à aucune image apprise dans les nœuds du graphe, alors un nouveau nœud est rajouté au graphe, c'est-à-dire, à la carte topologique. Une mauvaise détection de fermeture de boucle (faux positifs et faux négatifs) influe sur la précision des cartes. Les faux positifs créent d'une part, des arrêtes parasites entre les nœuds et peuvent conduire à des résultats indésirables lors de l'utilisation de la carte topologique. Les faux négatifs d'autre part, ne sont pas aussi dommageable mais peuvent entraîner un graphe dont les nœuds sont redondants. Par conséquent, une approche de détection de fermeture de boucle doit viser à produire zéro faux positif et à minimiser le nombre de faux négatifs.

1.6 MISE EN CORRESPONDANCE D'IMAGES

La mise en correspondance est très utilisée dans le domaine de la vision par ordinateur. En effet, c'est l'une des étapes les plus importantes lors de la reconstruction 3D à par-

tir d'une paire d'images stéréo ou d'une séquence d'images ou lors de la cartographie d'un environnement. C'est pour cette raison que beaucoup de chercheurs se sont intéressés à ce sujet durant les dernières décennies. Malgré tout, la mise en correspondance demeure une tâche difficile et un sujet de recherche toujours ouvert. Les méthodes d'appariement classiques sont basées sur des grandeurs géométriques et/ou sur des grandeurs photométriques qui peuvent être soit des primitives globales ou locales.

La mise en correspondance est obtenue à partir des scores de similarité calculés sur les différentes primitives. Pour les primitives locales, une détection de ces primitives visuelles dans les images est faite d'abord, puis un descripteur local est calculé pour chacune des primitives détectées. Ce descripteur local permettra de calculer un score de similarité entre deux primitives. Finalement, l'appariement ou la mise en correspondance est réalisé(e) par un algorithme qui utilise ces scores et des contraintes géométriques additionnelles (Voir Figure 1.6).

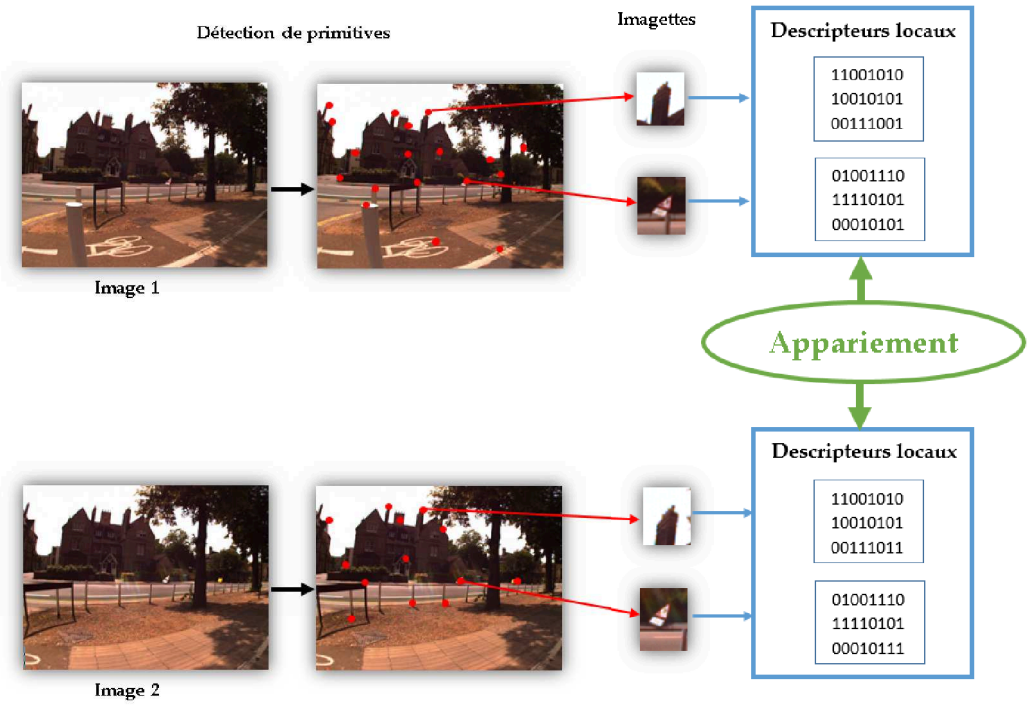


FIGURE 1.6: Appariement de primitives : détection, calcul de descripteurs locaux, appariement.

1.6.1 *Primitives globales*

Le but des primitives globales est de décrire l'image par une signature unique qui peut être considérée comme un vecteur de valeurs numériques. Ces vecteurs compacts servent à l'indexation d'image et à leur récupération lors d'une recherche de correspondance. Plusieurs primitives globales de l'image ont été proposées au cours des deux dernières décennies et ont été utilisées avec succès dans des applications de localisation et de détection de fermeture de boucle.

Les histogrammes globaux de couleurs calculés dans les espaces Rouge, Vert et Bleu (RVB) et Hue Saturation Value (HSV) sont utilisés pour faire de la localisation dans

des cartes topologiques [184]. La localisation est effectuée par l'apprentissage du plus proche voisin ou bien d'autres méthodes de matching d'histogrammes (Earth Mover Distance (EMD), Generalized Method of Moments (GMM) . . .).

Une représentation par valeurs propres des images a été utilisée dans le travail de Jogan and Leonardis (1999) [88] pour une localisation dans un ensemble d'images. De même, Kröse et al. (2001) [101] adoptent une réduction de dimension utilisant l'Analyse en Composantes Principales (ACP) dans un cadre probabiliste pour la localisation de robots dans un ensemble d'images.

Les coefficients de Fourier sont utilisés comme des signatures pour la localisation dans les cartes topologiques [120]. Il a été montré que ces signatures, qui sont simples à calculer sont suffisantes pour atteindre une bonne précision pour la localisation.

Un autre descripteur global d'image très populaire est le « The central idea - the essence » (GIST) [147, 148] qui vise à produire une représentation dimensionnelle réduite d'une image. Un ensemble de dimensions perceptuelles qui représentent la structure spatiale d'une scène est estimé en utilisant l'information spectrale et une localisation grossière. Par conséquent, l'image d'entrée est segmentée en une grille de 4×4 sur laquelle les histogrammes d'orientation sont extraits et représentés dans un seul vecteur. GIST a été utilisé avec succès dans la recherche et la récupération d'images [112] et la recherche d'images sur le Web [177].

Tous les descripteurs d'image globaux présentés ci-dessus ne sont pas robustes vis-à-vis des variations d'éclairage, des occultations et des grands changements de point de vue [126].

1.6.2 *Primitives locales d'image*

L'extraction de primitives locales de l'image repose sur l'utilisation d'un détecteur qui est en charge de trouver ces primitives dans l'image suivant un certain nombre de critères. Une fois ces primitives détectées, elles sont associées à un descripteur qui décrit l'information de la primitive.

On peut chercher à détecter différents types de primitives : des régions, des contours ou des points. Ces derniers sont les plus exploités, car d'une part les algorithmes géométriques tels que le calcul de pose, de géométrie épipolaire ou l'ajustement de faisceaux utilisent le plus souvent des points. D'autre part, la détection des points est moins sensible aux occultations que d'autres types de primitives couvrant une zone significative de l'image.

Contrairement aux primitives globales, les primitives locales sont généralement robustes aux occlusions, les variations d'éclairage, les différents points de vue, la rotation et les changements d'échelle (jusqu'à un certain degré). Différents types de primitives locales sont disponibles dans la littérature :

POINT (COINS) : Détecteur de Harris [68] , détecteur Harris-Laplace, détecteur Harris-Affine [126], Smallest Univalue Segment Assimilating Nucleus (SUSAN) [172], Features from Accelerated Segment Test (FAST) [178].

BLOB : Dans le domaine de la vision par ordinateur, la détection de blob se réfère à des méthodes mathématiques qui visent à détecter dans une image les zones qui diffèrent dans leurs propriétés, telles que la luminosité ou la couleur, par rapport aux zones environnantes. Détecteur Hessian [17], Hessian-Laplace/Affine [126], Scale Invariant Feature Transform (SIFT) [115], Speeded Up Robust Features (SURF) [14].

RÉGION : Intensity Based Regions (IBR) [182], Maximally Stable Extensible Regions (MSER) [119] et les Régions Saillantes (voir chapitre 2).

Le Tableau 1.1 regroupe les différentes primitives mentionnées ci-dessus et indique si elles sont invariants à la rotation, à l'échelle et plus généralement aux transformations affines.

DÉTECTEUR	COINS	BLOBS	RÉGIONS	INV ROTATION	INV L'ÉCHELLE	INV TRANSF AFFINES
Harris	✓	-	-	✓	-	-
Hessian	-	✓	-	✓	-	-
SUSAN	✓	-	-	✓	-	-
Harris-Laplace	✓	✓	-	✓	✓	-
Hessian-Laplace	✓	✓	-	✓	✓	-
SIFT	✓	✓	-	✓	✓	-
SURF	✓	✓	-	✓	✓	-
Harris-Affine	✓	✓	-	✓	✓	✓
Hessian-Affine	✓	✓	-	✓	✓	✓
Régions Saillantes	✓	✓	-	✓	✓	✓
Bordures	✓	-	-	✓	✓	✓
MSER	-	-	✓	✓	✓	✓

TABLE 1.1: Synthèse des détecteurs classiques [181].

Le choix de telles ou telles primitives dépend de l'application souhaitée. Par exemple, les coins de Harris et SUSAN sont invariants à la rotation et à la translation des images ; Harris-Laplace et Hessain-Laplace sont quant à eux invariants aux changements d'échelle alors que Harris-Affine, Hessian-Affine et MSER sont invariants aux transformations affines ; les primitives SIFT sont invariantes aux changement : d'échelle uniforme et d'orientation, et partiellement invariant lors de distorsions affines et changements d'éclairage ; les primitives SURF sont invariantes à la rotation et changement d'échelle, et possèdent un taux de répétabilité élevé et des caractéristiques distinctives. Plus de détails sur les détecteurs de primitives peuvent être trouvés dans [181].

Malgré leur robustesse, les primitives locales ne sont pas aussi compactes que les descripteurs globaux. Il peut y avoir des centaines, voire des milliers de primitives locales détectées sur chaque image ; et pour des applications comme la recherche d'image cela peut conduire à un véritable goulot d'étranglement de calcul. Par exemple, en considérant une image avec 500 primitives de type SIFT, chacune de dimension égale à 128, l'image doit être mise en correspondance avec une autre ayant un nombre égale de primitives ; une approche brute va prendre $500 \times 500 \times 128$ opérations tandis que l'approche du plus proche voisin peut être relativement efficace, mais nécessitera plus de mémoire ; si ce même processus doit être répété pour une application de recherche d'images sur une grande base de données d'images, le temps de calcul sera énorme et augmentera avec le nombre d'images se trouvant dans la base de données.

1.6.3 *Calcul de descripteurs locaux*

Chaque primitive détectée peut être décrite à l'aide d'un descripteur extrait de l'imagette qui est définie comme étant la région centrée autour de l'emplacement de la primitive. Le descripteur local est un condensé de l'information présente dans l'imagette. L'invariance aux changements de luminosité et de point de vue, sont parmi les qualités souhaitées pour un descripteur local. Mais une plus grande invariance va être accompagnée d'un temps de calcul plus élevé.

Soit (I_1 et I_2) deux images et ($P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$) deux primitives que l'on a extrait de (I_1 et I_2), respectivement. Le descripteur local le plus simple est un vecteur contenant l'intensité de chaque pixel d'une imagette de taille fixe. Dans ce cas, un score qui mesure la ressemblance (similarité) entre deux primitives est la Sum of Squared Difference (SSD) somme des différences de niveau de gris pixel par pixel.

$$SSD(P_1, P_2) = \sum_{d \in V} (I_1(P_1 + d) - I_2(P_2 + d))^2 \tag{1.1}$$

avec :

$$V = \{-N, \ldots, N\} \times \{-N, \ldots, N\} \tag{1.2}$$

Cette façon de calculer un score de ressemblance entre deux imagettes a l'avantage d'être très simple et très rapide à calculer mais elle n'est pas invariante aux changements d'illumination. Pour cela, il est préférable d'utiliser une corrélation centrée normée, ou Zero-mean Normalized Cross Correlation (ZNCC) :

$$ZNCC(P_1 \mid P_2) = \frac{\sum_{d \in V} (I_1(P_1 + d) - \overline{I_1}(P_1))(I_2(P_2 + d) - \overline{I_2}(P_2))}{\sqrt{\sum_{d \in V} (I_1(P_1 + d) - \overline{I_1}(P_1))^2} \sqrt{\sum_{d \in V} (I_2(P_2 + d) - \overline{I_2}(P_2))^2}} \tag{1.3}$$

avec :

$$\begin{cases} \overline{I_i}(P_i)) & = & \frac{1}{|V|} \sum_{d \in V} (I_i(P_i + d)) \\ V & = & \{-N, \ldots, N\} \times \{-N, \ldots, N\} \end{cases} \tag{1.4}$$

Cette méthode demande aussi peu de temps de calcul, ce qui permet de traiter de nombreuses primitives (plusieurs centaines par image) en temps réel. Le descripteur local choisi ainsi que les méthodes de mesure de ressemblance sont sensibles aux changements de perspectives, de rotation et de point de vue.

Pour pallier à ces inconvénients, d'autres méthodes peuvent être utilisées. Certes souvent plus coûteuses en temps de calcul, certaines d'entre elles sont utilisées dans des applications de vision temps réel, comme l'approche proposée par Lepetit and Fua (2006) [109] qui consiste à considérer le problème de mise en correspondance de primitives comme un problème de classification.

D'autres descripteurs locaux peuvent être aussi utilisés, en particulier ceux qui sont invariants à des transformations géométriques plus complexes. Les types les plus

communs de descripteurs sont SIFT, SURF, BRIEF, Orientation BRIEF (ORB) et Binary Robust Invariant Scalable Key-points (BRISK). En outre, la nature binaire de ces deux derniers descripteurs facilite de façon efficace la mise en correspondance de primitives et la quantification basée sur la distance de Hamming. A part l'avantage de calcul, les descripteurs binaires ne sont pas aussi robustes que les descripteurs SIFT ou SURF.

1.6.4 *Algorithmes d'appariement*

Un algorithme d'appariement va constituer une liste de couples de points à partir d'une liste de candidats potentiels. Chaque candidat est représenté par un triplet (P1, P2, s) où « P1 » est une primitive dans la première image, « P2 » une primitive dans la seconde image et « s » le score de similarité ou de ressemblance calculé en utilisant les descripteurs locaux de « P1 » et « P2 ».

Pour obtenir une liste de couples définitive plusieurs stratégies peuvent être envisagées. Souvent, la minimisation du nombre de candidats est faite par élimination des candidats ayant un score inférieur ou supérieur à un certain seuil. D'autres stratégies plus simples consistent à soit garder les meilleurs soit à éliminer les plus mauvais [163].

Il existe aussi d'autres méthodes exploitant des contraintes de voisinage, c'est-à-dire, des contraintes géométriques, car il est naturel de penser que deux points voisins dans l'image « I_1 » vont probablement être mis en correspondance à deux points voisins aussi dans l'image « I_2 ». Au final, chaque élément de la liste de couples doit respecter la contrainte d'unicité, c'est-à-dire, chaque point de l'image « I_1 » doit être apparié à au plus un point de l'image « I_2 ».

1.7 CONCLUSION

La richesse des informations contenues dans une image a permis à la vision d'être un élément important lors de développement d'applications robotiques autonomes. Néanmoins, les artefacts potentiellement présents dans les images rendent la tâche de reconnaissance et d'interprétation de l'information visuelle extrêmement compliquée. Ce qui peut enduire à un résultat imprécis (au sens des erreurs de localisation) et incertain (au sens des possibles ambiguïtés, des erreurs de reconnaissance). Il est de ce fait, très important d'utiliser des primitives robustes, stables et ayant un taux de répétabilité élevé afin d'obtenir de bonnes performances au niveau de la mise en correspondance et de la localisation. C'est pour ces raisons que les régions saillantes seront exploitées dans la suite du manuscrit. Ces primitives sont introduites dans le chapitre suivant.

2

SAILLANCE VISUELLE ET SON UTILISATION EN ROBOTIQUE NAVIGATION

2.1 INTRODUCTION

Actuellement les robots se confrontent à des défis similaires aux humains, c'est à dire qu'ils doivent faire face à une grande quantité de données en entrée et sélectionner les éléments les plus prometteurs uniquement. La *saillance visuelle* est l'un des principaux mécanismes de la perception qui permet à l'homme de sélectionner efficacement les données visuelles qui ont le plus d'intérêt. La première définition de l'attention visuelle fut donnée par le père de la psychologie, William (1890) [193]. Sa définition, dans sa version anglaise originale, est la suivante :

« *Everyone knows what attention is. It is the taking possession of the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalization, concentration of consciousness are of its essence . . .* »

« *L'attention est la prise de possession par l'esprit, sous une forme claire et vive, d'un objet ou d'une suite de pensées parmi plusieurs qui semblent possibles . . . Elle implique le retrait de certains objets afin de traiter plus efficacement les autres* »

Un des rôles de ce mécanisme est d'accélérer le processus de vision, en réduisant sensiblement la quantité d'informations visuelles qui sera traitée par les tâches de plus haut niveau. La réduction se fait par la suppression d'informations redondantes et par la sélection rapide des informations les plus pertinentes en se focalisant sur les éléments les plus saillants. Un élément visuellement saillant, est un élément qui ressort prioritairement lors de la perception visuelle d'une scène, au point de prendre une importance cognitive particulière.

S'il est évident que l'attention visuelle est un concept utile pour les humains, il est aussi un concept intéressant pour les robots mobiles autonomes où le traitement en temps réel est nécessaire et où une grande base de données d'image doit être traitée, car le mécanisme d'attention visuelle permet de gérer la complexité des entrées perceptuelles en favorisant des régions d'image pour le traitement ultérieur.

Comme les robots opèrent généralement dans les mêmes environnements que les humains, il est raisonnable d'imiter le système d'attention humain pour accomplir ces tâches. En outre, dans les domaines de l'interaction homme-robot, il est utile de créer un point commun d'attention entre l'homme et le robot pour s'assurer que les deux communiquent sur le même objet. Avoir des mécanismes similaires pour les deux (hommes et robots) facilite cette tâche.

Les principales sources d'inspiration des chercheurs en vision artificielle étaient les différentes études réalisées sur l'attention visuelle dans des domaines divers tels

que la psychologie, la psychophysique et la neurophysiologie. Plusieurs modèles informatiques, se basant essentiellement sur la théorie d'intégration des primitives [180], se sont succédés comme celui présenté par Koch and Ullman (1987) [98]. Des études plus poussées ont été élaborées par Itti et al. (1998) [84] par la suite pour la réalisation d'un modèle d'attention visuelle basé sur la saillance, qui sera le modèle de base pour un certain nombre des modélisations informatiques suivantes comme celle de Itti and Koch (2000) [82] ; Ouerhani (2004) [151] et Frintrop (2006) [56].

Un des objectifs de ces modèles informatiques est de réduire le coût computationnel des tâches de plus haut niveau, en réduisant l'information visuelle à son « *essence* » [160]. Cependant, la multiplicité des primitives, et le caractère multi-échelle des traitements impliquent une charge de calcul importante sur le bas-niveau.

2.2 L'ATTENTION VISUELLE HUMAINE

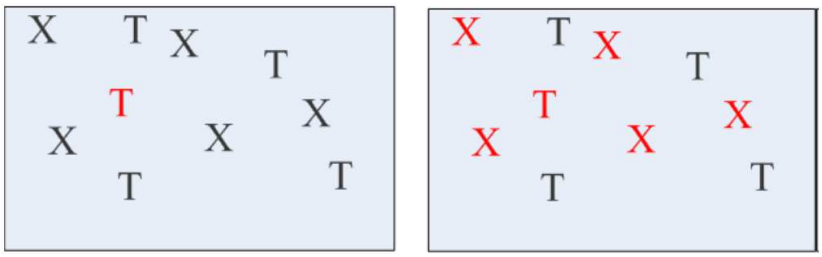
L'attention visuelle humaine désigne le mécanisme de sélection des informations visuelles spatio-temporelles du monde visible. Étant donné que l'environnement visuel contient beaucoup d'informations. Le système visuel humain étant de plus intrinsèquement limité en capacité de traitement, ce dernier s'est adapté par le biais du mécanisme de l'attention visuelle pour réduire la quantité d'information à traiter et pour ne conserver que les informations les plus importantes. En d'autres termes, l'attention visuelle permet d'utiliser de façon optimisée les ressources biologiques ; ainsi, seule une petite partie des informations incidentes est transmise aux aires supérieures du cerveau [11].

William (1890) [193] et Nakayama and Mackeben (1989) [138] avaient émis l'hypothèse d'au moins deux modèles de mécanismes de l'attention, un modèle d'attention ascendant (Bottom-Up), et un modèle d'attention descendant (Top-Down). Ces deux modèles peuvent être décrits comme suit :

LES MODÈLES ASCENDANTS (BOTTOM-UP) : Dans ces modèles, l'analyse des éléments se fait à partir des propriétés de la scène perçue. Ce sont des processus automatiques sélectionnant les informations visuelles selon leur saillance. Ce mécanisme de sélection se fait donc sans aucune connaissance a priori sur la scène. En effet, même en l'absence de tâche à effectuer, le regard se balade lors de l'observation d'une scène et décrit un parcours que ces modèles s'attachent à analyser et à définir (Figure 2.1a).

LES MODÈLES DESCENDANTS (TOP-DOWN) : Ce sont des processus contrôlés. Dans certains cas c'est le cerveau lui-même qui envoie directement l'information vers les systèmes sensoriels. Ainsi les modèles d'attention Top-Down sont des modèles principalement dirigés par les connaissances a priori sur la scène (Figure 2.1b). En d'autres termes, ces mécanismes sont pilotés par la tâche à effectuer (*Task-dependent*). La Figure 2.2 donne, pour un même observateur et pour cinq taches différentes à effectuer, les stratégies visuelles associées à la tâche.

Ces deux processus sont indispensables pour pouvoir interpréter des scènes en temps réel. Le mécanisme étudié dans ce présent mémoire est le « Bottom-Up ». En effet, afin de réduire la complexité du processus de mise en correspondance des images, et augmenter la robustesse et l'efficacité de ce processus, l'extraction dans une image d'informations pertinentes et robustes, vis-à-vis des éventuels changements comme le



(a) Trouver la lettre rouge. (b) Trouver la lettre « T » rouge.

FIGURE 2.1: Trouvez le « T » rouge, (a) impossible de ne pas le voir : processus Bottom-Up (cas disjonctif = traitement parallèle) ; (b) Plus difficile qu'en (a) car cela demande un processus actif mettant en jeu diverses « stratégies » perceptives : processus Top-Down (cas conjonctif = traitement série) [105].

point de vue ou les conditions d'acquisition, sans aucune connaissance préalable est souhaitée.

La théorie de l'intégration de primitives (Feature Integration Theory (FIT)) de Treisman and Gelade (1980) [180] doit être présentée pour mieux comprendre les deux mécanismes de l'attention visuelle, et plus particulièrement le mécanisme Bottom-Up. Les travaux de Treisman and Gelade (1980) [180] reposent sur des expériences de recherche visuelle, mesurant le temps de réaction nécessaire pour distinguer un objet cible enfoui parmi d'autres objets (*distracteurs*). Ces objets peuvent être caractérisés par une seule dimension visuelle (la couleur, l'orientation, la forme ...), ou plusieurs à la fois. Les résultats de ces expériences ont dévoilé deux comportements distincts :

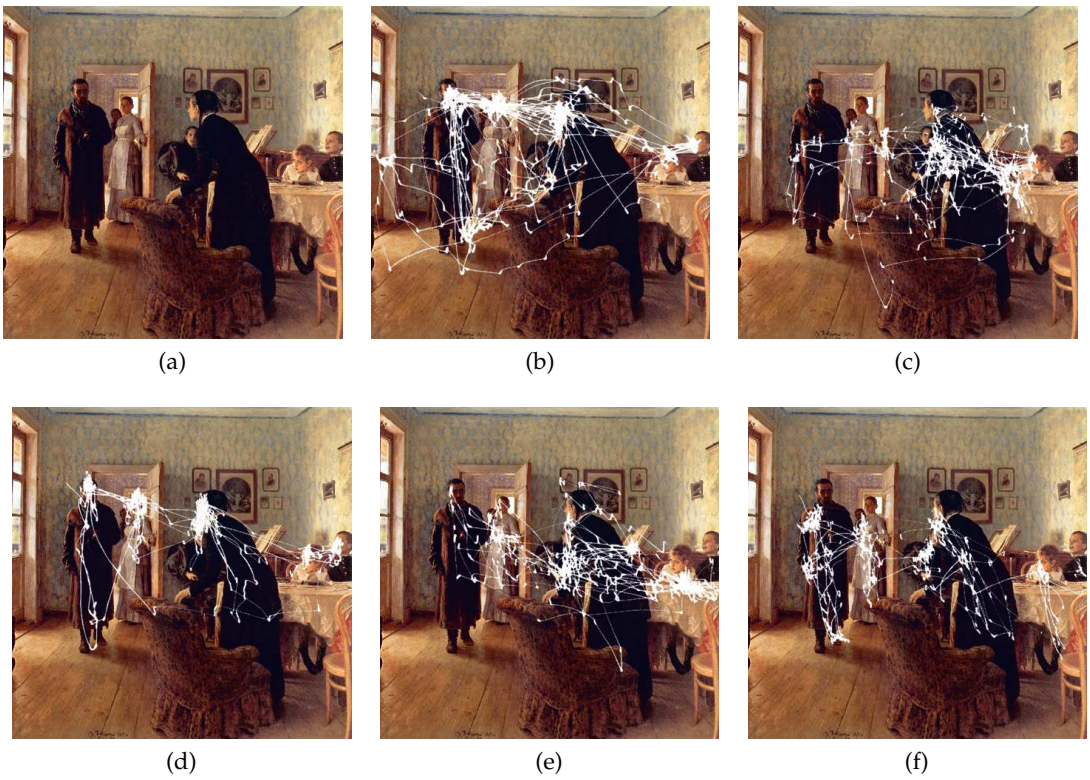
- Un comportement disjonctif, lorsque la cible diffère des distracteurs par au moins une caractéristique visuelle (exemple de la Figure 2.1a), dans ce cas-là le temps de réaction nécessaire pour trouver la cible est constant quel que soit le nombre de distracteurs. La cible saute aux yeux (dans la littérature scientifique, on parle de phénomène de « *pop-out* » issu du verbe anglais associé)

- Un comportement conjonctif, lorsque la cible est une combinaison de caractéristiques visuelles (voir Figure 2.1b), dans ce cas-là le temps de réaction augmente linéairement avec le nombre de distracteurs, car afin de trouver la cible tous les objets doivent être examinés séquentiellement.

Ainsi, le comportement disjonctif est à rapprocher du mécanisme Bottom-Up qui, finalement, permet de traiter les caractéristiques visuelles d'une scène rapidement et d'une façon massivement parallèle. Tandis que le comportement conjonctif, quant à lui, est à rapprocher du mécanisme Top-Down qui est un mécanisme lent et traitant les informations visuelles de façon séquentielle [105].

2.2.1 *Le mécanisme inhibiteur de l'attention*

La fonction première de l'attention visuelle sélective est de diriger le regard vers des objets d'intérêt contenus dans un environnement visuel général. Ce mécanisme de l'attention visuelle est constitué de deux composantes comme vu précédemment : Top-Down et Bottom-Up.



(a) (b) (c)

(d) (e) (f)

FIGURE 2.2: Impact de la tâche à effectuer sur le trajet oculaire lors de la visualisation d'une image par le même observateur : (a) image originale (tableau de Ilya Repin 1884 intitulé « *An Unexpected Visitor* ») ; (b) aucune tâche n'est donnée (Free-viewing mode) ; (c) première tâche : estimer le niveau social des personnages ; (d) deuxième tâche : évaluer leur âge ; (e) troisième tâche : que faisaient les personnages avant l'arrivée du visiteur ? ; (f) quatrième tâche : souvenez-vous des habits portés par les différents personnages (Extrait de [198]).

En plus de l'influence de ces deux composantes un autre mécanisme est à l'œuvre, appelé inhibition de retour (Inhibition Of Return (IOR)), qui consiste à inhiber une zone inspectée afin d'éviter que l'attention visuelle se porte continuellement sur la même zone. Ainsi grâce au mécanisme d'attention visuelle et l'IOR, les différentes régions saillantes d'une scène sont explorées séquentiellement, une par une, ce qui facilite le processus d'inspection visuelle. Par exemple, dans le cadre d'une recherche visuelle conjonctive, de type séquentielle, l'inhibition de retour est primordiale puisqu'elle évite de continuellement revérifier les mêmes objets [96, 97].

2.2.2 *Les caractéristiques attirant l'attention visuelle*

L'être humain possède une attention visuelle sélective ce qui veux dire que son système visuel répond de façon particulière à un certain nombre de signaux provenant des objets et à des évènements de l'environnement. D'après Yantis and Jonides (1990) [197] c'est l'apparition soudaine d'un objet dans une scène qui attire plus et de façon intuitive l'attention visuelle. Lorsque l'objet est incohérent avec la scène, Henderson and Hollingworth (1999) [72], [71] montrent que les observateurs ont tendance à faire des fixations plus longues et plus fréquentes sur cet objet saillant.

Afin de ressortir les degrés de saillances des différents éléments d'une scène (objet ou événement), il possible d'en extraire les primitives pré-attentives qui la caractérisent, puis rassembler les différents degrés de saillance de chaque caractéristique par un système de combinaison des primitives. Parmi les primitives pré-attentives les plus utilisées, il y a : l'intensité, la couleur (teinte) et l'orientation ... Il existe aussi de nombreuses autres caractéristiques attirant l'attention, par exemple, la taille, la courbure, et le mouvement [179].

2.2.3 *Combinaison des primitives*

Après l'extraction des différentes primitives pré-attentives, elles sont intégrées soit par un système de vote, soit par des techniques de sommation naïve ou pondérée pour donner une carte de saillance finale.

Un élément d'une scène peut avoir un degré de saillance bas sur l'une des primitives, et élevé sur une autre. L'intérêt de la combinaison, est d'un côté de compenser le degré de saillance dans la primitive à faible réponse, et d'un autre côté d'atténuer le degré de saillance correspondant à une forte réponse (voir Figure 2.3). A partir de ce formalisme, le degré de saillance accordé à un élément de la carte finale est d'autant plus élevé que le nombre de primitives pour lesquelles l'élément ressort prioritairement est grand.

Dans un processus de vision, le résultat de l'observation attentionnelle d'une scène va différer selon le type de cette observation, c'est à dire selon la primitive considérée. Le mécanisme de combinaison des primitives a comme but de donner aux éléments de la scène observée différents degrés de saillance relative à l'ensemble des primitives choisies.

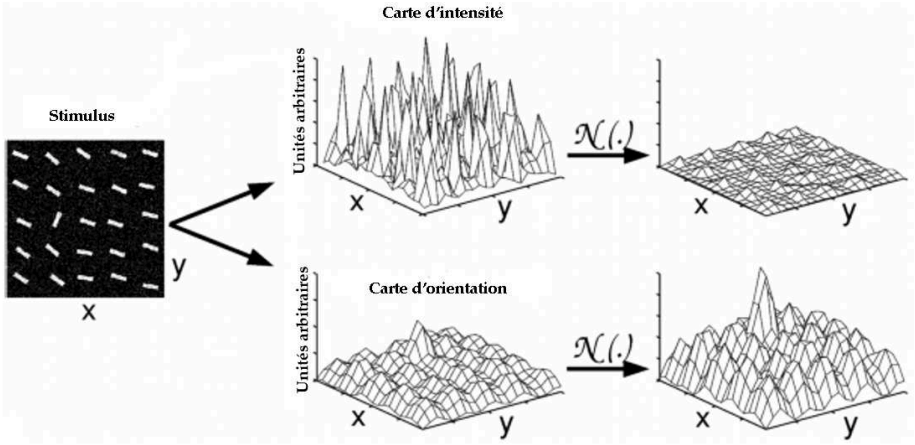


FIGURE 2.3: Si on considère la primitive intensité seulement, on a un niveau de saillance uniforme qui ne fait pas apparaître des points visiblement saillants. Par contre, si on ajoute la primitive orientation, on aura une apparition d'un niveau de saillance non uniforme ; ce qui nous donnera des points visiblement saillants [84].

2.3 MODÈLES COMPUTATIONNELS D'ATTENTION VISUELLE

Un système computationnel d'attention modélise les principes de l'attention sélective humaine et vise à sélectionner la partie des données d'entrée sensorielle qui est la plus prometteuse pour un traitement plus avancé.

Depuis une vingtaine d'années, de nombreuses études ont été réalisées pour trouver un modèle informatique de l'attention visuelle. Les systèmes d'attention visuelle présentés dans cette section sont inspirés par les concepts du système visuel humain, mais sont conçus avec un objectif d'ingénierie, qui signifie que leur but est d'améliorer les systèmes de vision dans des applications techniques.

2.3.1 *Modèles empiriques et statistiques*

2.3.1.1 *Modèles empiriques*

Un modèle empirique est un modèle éloigné des propriétés du système visuel humain et qui se base principalement sur des méthodes de traitement d'images. Parmi les modèles les plus connus est celui de Osberger and Maeder (1998) [150] qui déterminent la carte de saillance d'une scène en effectuant une segmentation en régions homogènes suivie d'une taxonomie des régions se basant sur certains critères, (la taille, le contraste, la forme ...).

De même Luo and Singhal (2000) [116] définissent un ensemble d'éléments visuels susceptibles d'attirer l'attention tel que la couleur, la texture et la forme. Ces derniers sont extraits afin de piloter une segmentation favorisant l'apparition de régions d'intérêt.

2.3.1.2 *Modèle statistique*

Le système d'attention visuelle est attiré par les objets (régions) différents de leurs voisinages, c'est-à-dire, en contraste avec leurs voisinages. Le modèle statistique le plus connu et qui utilise cette propriété du système visuel humain est celui proposé par Oliva et al. (2003) [149].

Oliva et al. (2003) [149] calculent la carte de saillance d'une scène en prenant l'inverse de la probabilité d'apparition des éléments la constituant. Oliva et al. (2003) [149] ont montré dans leur article que les performances de leur modèle sont comparables à celles du modèle de Itti et al. (1998) [84] décrit dans la section suivante.

2.3.2 *Modèles psycho-visuel*

2.3.2.1 *Modèle fondateur*

Le modèle fondateur de Koch and Ullman (1987) [98] simule un mécanisme Bottom-Up totalement guidée par les données, basé sur une architecture biologiquement plausible,

et qui ne prend en compte aucune connaissance préliminaire sur la scène. La Figure 2.4 présente cette architecture devenue par la suite une référence.

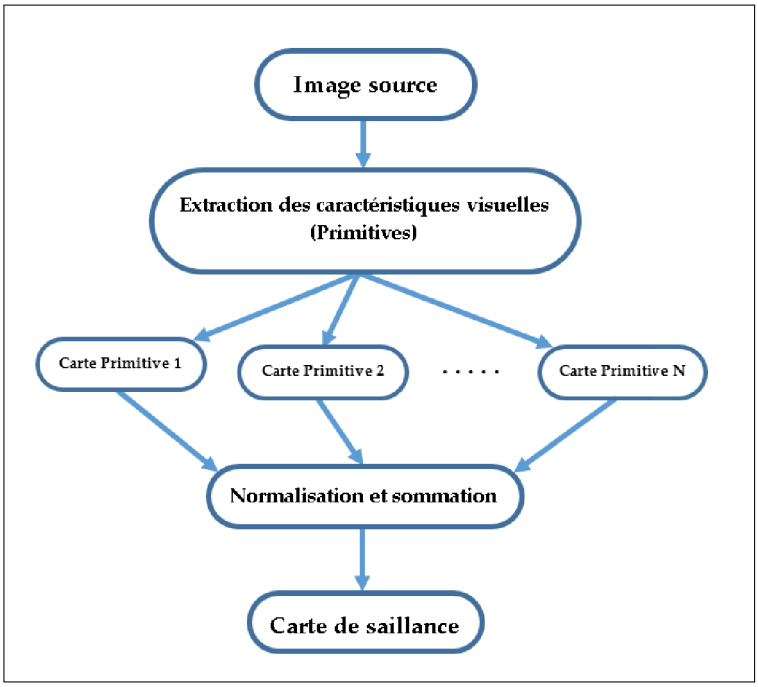


FIGURE 2.4: Architecture biologiquement plausible d'un modèle d'attention visuelle proposée par Koch and Ullman (1987) [98].

Quand il a été publié, le modèle n'avait pas encore été implémenté, mais il a fourni le raisonnement algorithmique servant de base pour les implémentations ultérieures et pour de nombreux modèles de calcul actuels d'attention visuelle.

Ce modèle commence par l'extraction d'un certain nombre de caractéristiques de la scène. Il n'y a pas vraiment de liste exhaustive de ces caractéristiques mais il existe un consensus au sujet d'un certain nombre d'entre elles [195] : le contraste, l'orientation, la couleur, et la direction du mouvement ... Chaque caractéristique extraite donne naissance à *une carte d'évidence*, qui donne les points importants d'une scène suivant chaque caractéristique. Cette notation a été introduite par Milanese (1993) [127] pour désigner la saillance d'une caractéristique.

La définition originelle donnée par Koch and Ullman (1987) [98] est la suivante : « *la carte de saillance est une représentation de l'environnement accentuant les régions d'intérêt du champ visuel* ». La carte de saillance finale, qui encode la saillance ou le pouvoir attracteur pour toutes les positions de la scène visuelle, s'obtient en combinant les différentes cartes d'évidences. Enfin, le système d'attention balaye simplement cette carte dans un ordre décroissant de saillance.

Une contribution importante du travail de Koch and Ullman (1987) [98] est l'utilisation d'un réseau de neurones reproduisant de manière distribuée un mécanisme de type Winner-Takes-All (WTA) afin de déterminer la région la plus saillante dans une carte de saillance. Une description détaillée de la mise en œuvre de ce mécanisme est donnée dans l'article [98]. Cette approche montre comment un tel mécanisme pourrait être implémenté dans le cerveau humain. Cependant, pour un système artificiel, une

telle implémentation d'un WTA est certainement une surcharge car il existe des moyens beaucoup plus faciles permettant de trouver le maximum d'une carte de saillance [56].

Enfin, les auteurs suggèrent un mécanisme d'inhibition de la région sélectionnée (IOR) provoquant un déplacement automatique vers la prochaine région la plus saillante (voir sous-section 2.2.1).

Pour résumer, ce modèle comporte trois étapes principales (voir Figure 2.4). Tout d'abord les cartes de caractéristiques sont extraites, puis les cartes d'évidence (ou cartes de caractéristiques), et pour finir la carte de saillance. L'architecture proposée est exclusivement Bottom-Up ; il n'est pas discuté la façon avec laquelle des processus Top-Down peuvent contribuer à la sélection des régions saillantes.

2.3.2.2 *Modèle NVT*

Le modèle de Itti et al. (1998) [84] (NVT), qui dérive du modèle de Koch and Ullman (1987) [98], est actuellement l'un des modèles d'attention les plus connus dû au nombre de parutions scientifiques associées ([84, 82, 85, 83, 139]) et au fait que les codes sources du modèle soient disponibles sur le net.

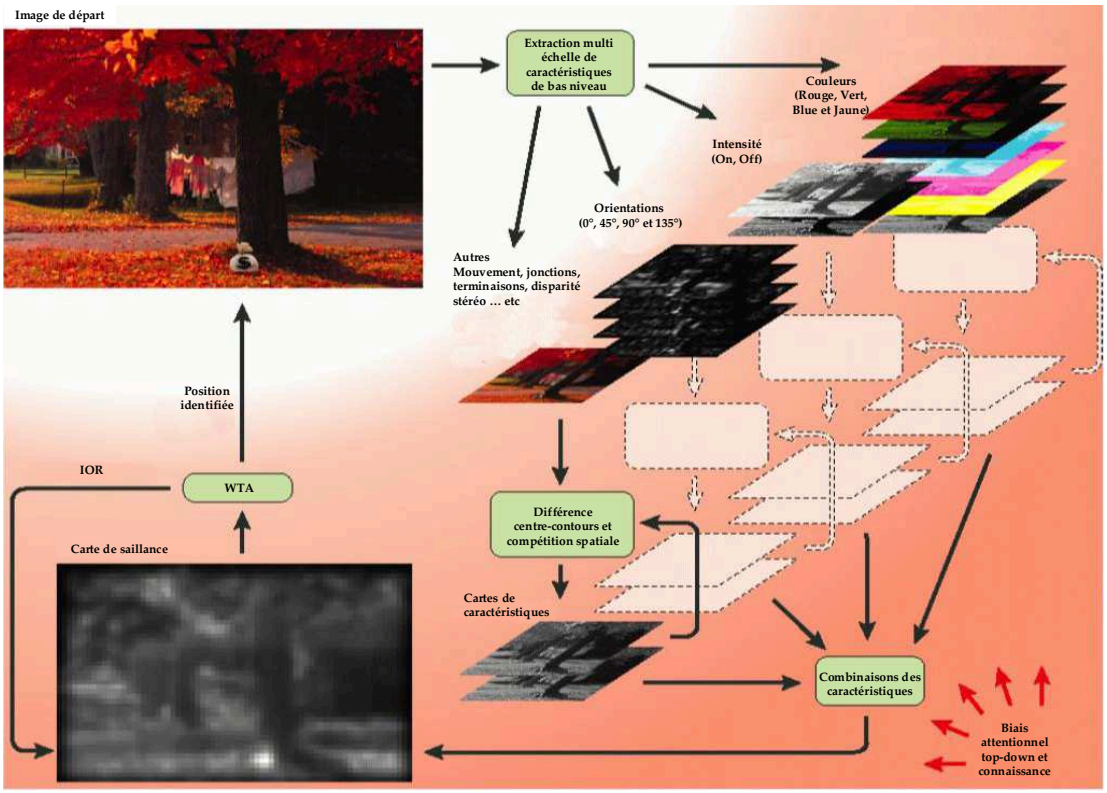


FIGURE 2.5: Le modèle de Itti and Koch (2001) [83].

La Figure 2.5 montre la structure de base du modèle. L'idée de cartes de primitives pré-attentive, de carte de saillance, de WTA et d'IOR ont été retenues du modèle Koch and Ullman (1987) [98].

A partir d'une image, trois caractéristiques sont calculées : couleur, intensité et orientation. Pour chaque caractéristique, une pyramide d'images est construite pour permettre de faire un traitement à différentes échelles. En traitement d'images, la *pyramide* est une représentation multi-résolution d'une image. Elle permet de modéliser l'image à différentes résolutions, depuis l'image initiale à une image très grossière, ce qui permet la détection de caractéristiques à différentes échelles.

Le processus « *Center-Surround* » ou « *Centre-Contour* » permet de déterminer les cartes d'évidences de chaque caractéristique, qui seront par la suite combinées pour donner la carte de saillance finale. Le nom du processus « *Centre-Contour* » a été donné pour la raison suivante : les neurones visuels sont généralement excités par une petite région de l'espace visuel (le *centre* de leur champ récepteur), alors que le signal présent autour (*contour*) inhibe la réponse neuronale (par exemple cellule ON-OFF). Exemple dans le cas d'une ligne horizontale entourée par des lignes verticales, la réponse en cet endroit sera plus grande que si elle était entourée par des lignes horizontales.

Pour déterminer le point le plus saillant vers lequel l'attention va se diriger dans la carte de saillance, le processus WTA est utilisé. La région entourant le point saillant est ensuite inhibée à l'aide de l'IOR, ce qui permettra au système de trouver le prochain point saillant.

Les principales contributions de ce travail sont des élaborations détaillées sur la réalisation de concepts théoriques, l'implémentation concrète du modèle et son utilisation sur des scènes artificielles et réels. En plus, ils proposent une fonction de pondération pour la combinaison des différentes cartes d'évidence en favorisant les cartes avec quelques pics (points saillants pour cette carte) par rapport aux cartes avec plusieurs pics.

Pour évaluer la qualité du modèle de [84] nommé NVT, une comparaison avec le comportement humain a été réalisée dans les travaux de Parkhurst et al. (2002) [155]. Les auteurs ont comparé les fixations obtenus avec le système NVT à celles obtenues avec le système visuel humain sur les mêmes scènes et ont trouvé une bonne correspondance, particulièrement sur les premières fixations (points saillants). Ils ont également trouvé que la cohérence était dépendante de la nature de la scène : pour des scènes fractales (synthèse d'image), la cohérence était plus élevée que pour des scènes naturelles. Ceci s'explique par l'influence des connaissances apriori sur le traitement humain des scènes naturelles (approche Top-Down), un aspect non traité dans le modèle NVT.

Une plate forme de test (Beobot) pour le système d'attention NVT a été présentée dans ([36, 79, 80]. Dans [79], il a été montré comment le traitement peut être distribué entre différents processeurs permettant un calcul parallèle rapide.

2.3.2.3 *Modèle VOCUS*

Le modèle Visual Object detection with a CompUtational attention System (VOCUS) proposé par Frintrop (2006) [56] dans son travail partage les principaux concepts avec les modèles d'attention visuelle standards, en particulier avec le modèle de Koch and Ullman (1987) [98]. L'implémentation est basée sur le modèle NVT de Itti et al. (1998) [84], qui est actuellement l'un des systèmes les plus connus de l'attention. Cependant, il y a plusieurs différences concernant les détails d'implémentation ainsi que les éléments de conception structurelle qui permettent des améliorations considérables de la perfor-

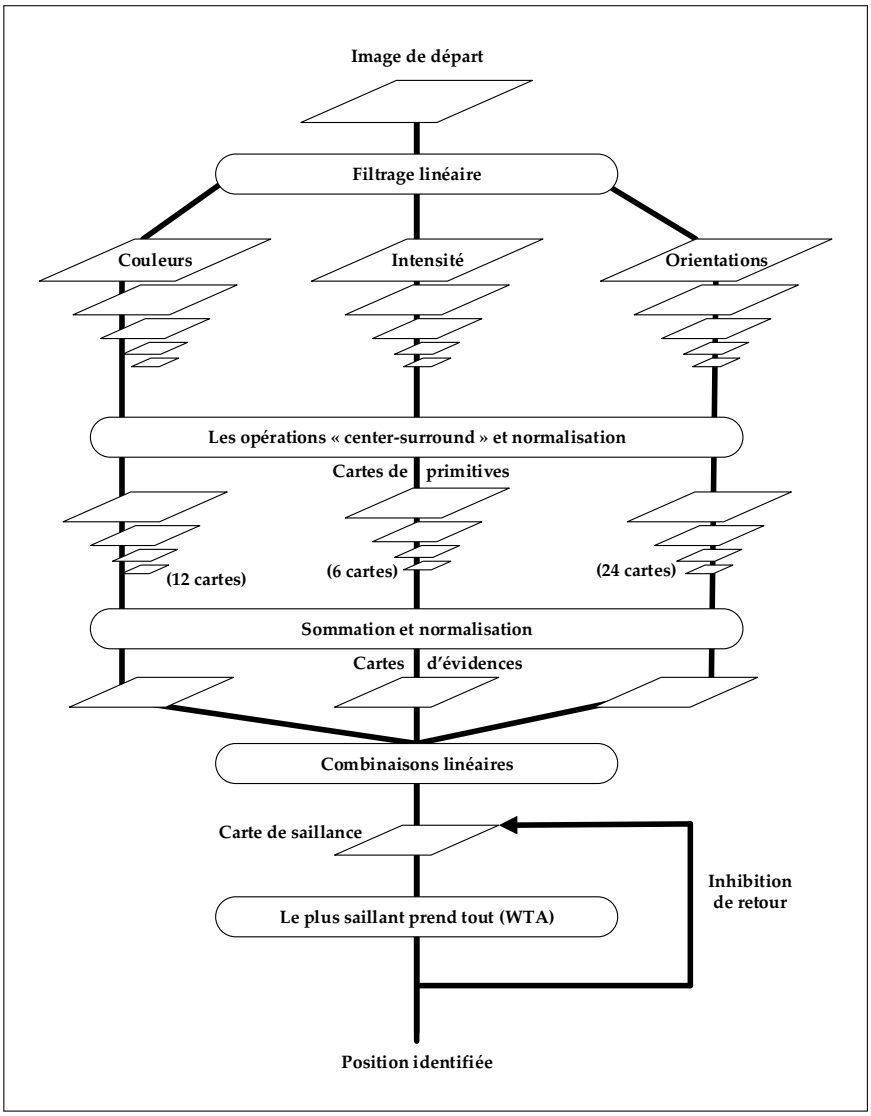


FIGURE 2.6: Le modèle NVT de Itti et al. (1998) [84].

mance. Comme c'est ce modèle qui a été choisi pour être utilisé dans les différents travaux présentés dans ce mémoire, la bonne compréhension du fonctionnement de ce modèle est nécessaire. C'est pour cela que les grandes étapes de ce modèle sont détaillées par la suite.

La structure du modèle d'attention visuelle (Bottom-Up) est représentée sur la Figure 2.7. Avant d'entrer dans les détails du système, un bref aperçu de la structure est présenté. Sur l'image d'entrée, trois cartes de caractéristiques différentes sont calculées : l'intensité, l'orientation et la couleur. Pour chaque dimension, les saillances sont calculées à différentes échelles et pour différentes caractéristiques, par exemple rouge, vert, bleu, et jaune pour la couleur.

Les cartes d'évidences de chaque caractéristique sont fusionnées en une seule carte de saillance. De cette carte, la région la plus saillante (Most Salient Region (MSR)) est extraite et le centre de l'attention (Focus of Attention (FOA)) est dirigé vers cette région. Après cela, le processus d'inhibition de retour inhibe la région sélectionnée dans la carte

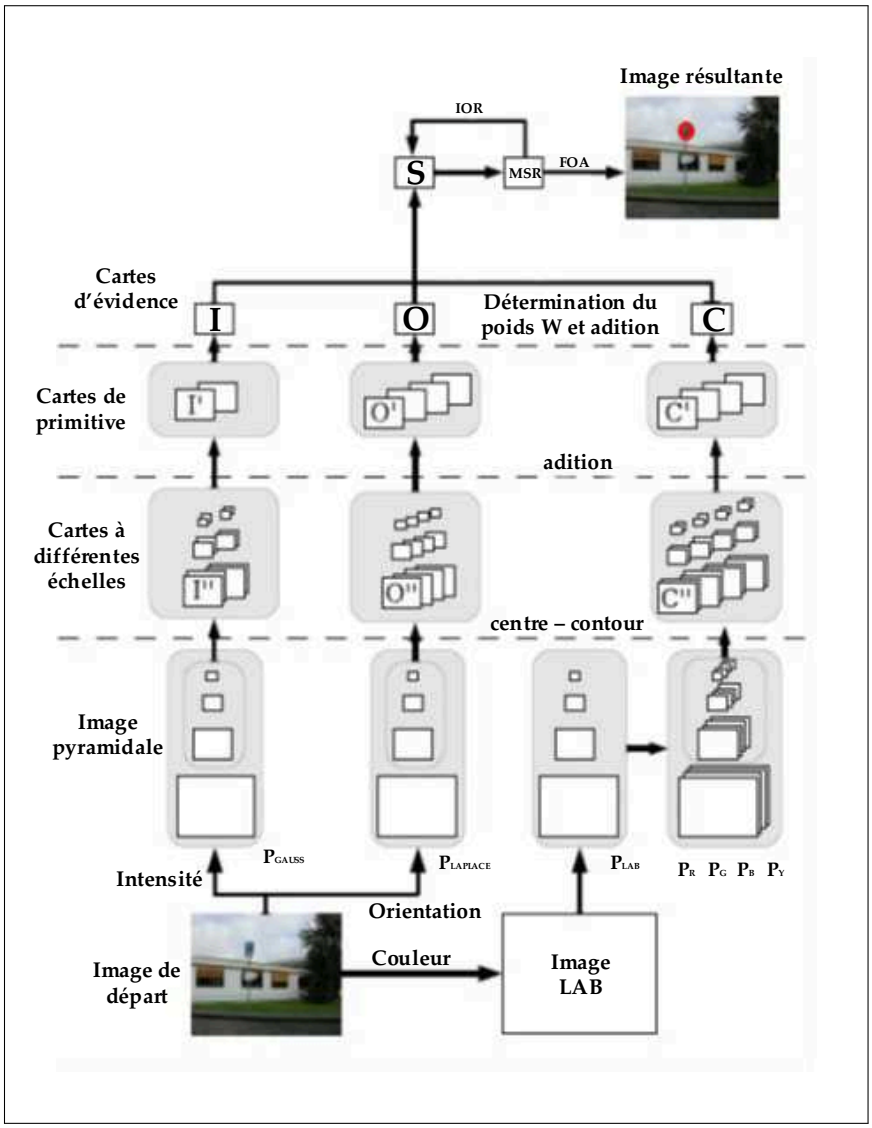


FIGURE 2.7: VOCUS - Le système de l'attention visuel Bottom-Up de Frintrop (2006) [56].

de saillance, ce qui permettra au système de trouver la prochaine région saillante. Ces deux dernières étapes sont répétées de manière itérative jusqu'à ce que un nombre suffisant de régions saillantes soient trouvées. Ce nombre de régions dépend de l'application et des performances que l'on veut obtenir (voir chapitre 3).

La première étape de ce modèle consiste à extraire certaines caractéristiques d'une image pour obtenir des cartes dites de caractéristiques, représentant l'image de départ projetée dans plusieurs espaces de caractéristiques bien définies. Ainsi une représentation multi-caractéristique de la scène est obtenue. Cette étape crée trois canaux à partir d'une image couleur RVB :

- Un canal lié à l'intensité I,
- Un canal couleur composé de quatre composantes : Rouge, Vert, Bleu et Jaune, générées à partir de l'image Lab

- Un canal orientation, qui est obtenu à partir du canal intensité.

La deuxième étape est une décomposition hiérarchique sur 9 niveaux via des pyramides Gaussiennes [29] effectuée sur chaque composante pour obtenir une représentation discrète multi-résolution de l'image. Cette représentation est donnée par une série d'images de plus en plus petites, qui s'obtiennent par un sous échantillonnage avec application d'un filtre Gaussien sur ces images, ce qui donne à la fin naissance à des formes pyramidales dyadique[1] (Voir Figure 2.8).

La troisième étape représente le mécanisme ou le processus « *Centre-Contour* » permettant l'extraction des informations pertinentes en contraste avec leur voisinage à partir des différents niveaux de la pyramide. Ceci est effectué par le calcul de la différence entre les niveaux fins et grossiers (Figure 2.8).

Dans le modèle de Frintrop (2006) [56] ce processus est décrit comme suit : le centre est un pixel au niveau $c \in \{2,3,4\}$, et le contour, c'est-à-dire, région contournant le pixel, correspondant au niveau $s = c + \sigma$ avec $\sigma \in \{3,4\}$, est déterminée par le calcul de la moyenne des pixels entourant le pixel central pour deux tailles différentes de fenêtre de rayon 3 et 7 respectivement. Les niveaux 0 et 1 sont trop précis et ne sont pas assez lisses pour être utilisé, de même les niveaux plus grands que 8 ne sont pas utilisés (s'ils existent) car ils ne sont pas assez significatifs. L'Algorithme 1 donne un résumé des opérations principales de l'algorithme *Centre-Contour*.

Dans ce modèle l'opérateur $\ominus$ représente l'opérateur de différenciation inter niveaux ; ce dernier réalise en fait une interpolation du niveau grossier afin de pouvoir faire une différence point à point entre deux cartes de même résolution correspondant à différents niveaux

$$L_i(x,y) = G_i(x,y) - \mathrm{Expand}(G_i(x,y)) \tag{2.1}$$

Avec : « Expand » représentant une interpolation bilinéaire.

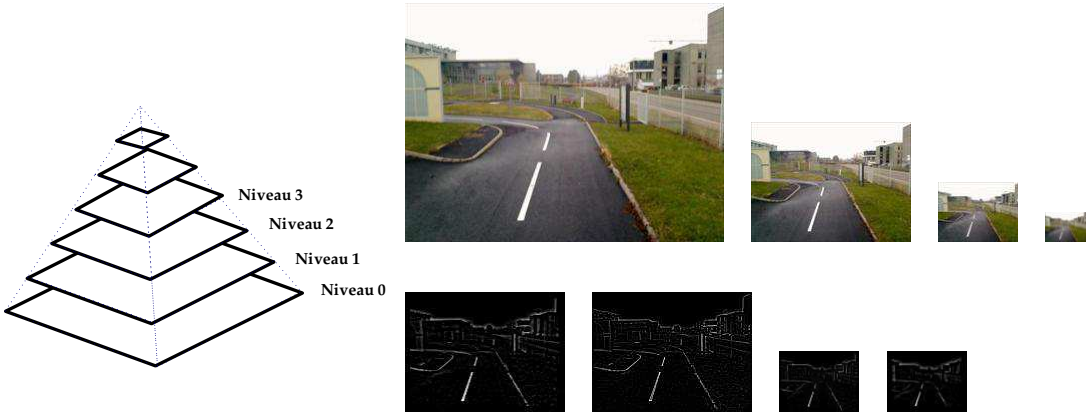


FIGURE 2.8: À gauche : structure pyramidale ; en haut à droite : exemple d'une pyramide dyadique ; en bas à droite : résultat obtenu avec l'opérateur *Centre-Contour* [27].

1 Une pyramide est dite dyadique si le passage d'un niveau (i) à un niveau $(i+1)$ se fait avec le rapport 1/4 sur les pixels.

Algorithme 1: L'algorithme *Centre-Contour* selon le modèle Frintrop (2006) [56] pour le calcul des cartes à différentes échelles.

1
$$
\begin{aligned}
&v(x,y) = \text{Pix}(x,y) \ \text{ du niveau } s \\
&\text{Centre} = v(x,y) \\
&\text{Contour} = \text{moyenne}(v(x-\sigma,y-\sigma),\ldots,v(x+\sigma,y+\sigma)) \\
&I_{(i,s,\sigma)}(x,y) = \begin{cases} \text{Centre} - \text{Contour} & \text{Si} \quad \text{Centre} > \text{Contour} \\ 0 & \text{Sinon} \quad . \end{cases}
\end{aligned}
$$

Pour le canal intensité, Soient (r,g,b) les canaux rouge, vert et bleu de l'image d'entrée. L'image d'intensité est obtenue avec l'équation $I = (r+g+b)/3$. A partir de I, une pyramide gaussienne $I(\sigma)$ avec $\sigma \in \{0\ldots8\}$ est créée. Le contraste d'intensité est calculé en appliquant l'algorithme *Centre-Contour* présenté dans le modèle de Frintrop (2006) [56] sur les niveaux s_2, s_3 et s_4, c'est à dire, sur les images lissées au moins deux fois car cela rend le système plus robuste au bruit [84]. Selon le système visuel humain, deux types de caractéristiques sont déterminés pour l'intensité : « *on-center* » qui répond fortement aux régions claires sur un fond sombre, et « *off-center* » qui répond fortement aux régions sombres sur un fond clair. Dans le modèle de Frintrop (2006) [56] ces deux types sont pris en considération et sont traités séparément ce qui donne au finale 12 cartes d'intensité $I_{(i,s,\sigma)}$ avec $i \in \{(on),(off)\}$, $s \in \{s_2,s_3,s_4\}$ et $\sigma \in 3,7$. La Figure 2.9 présente un exemple de résultats obtenus lorsque l'Algorithme 1 est utilisé.

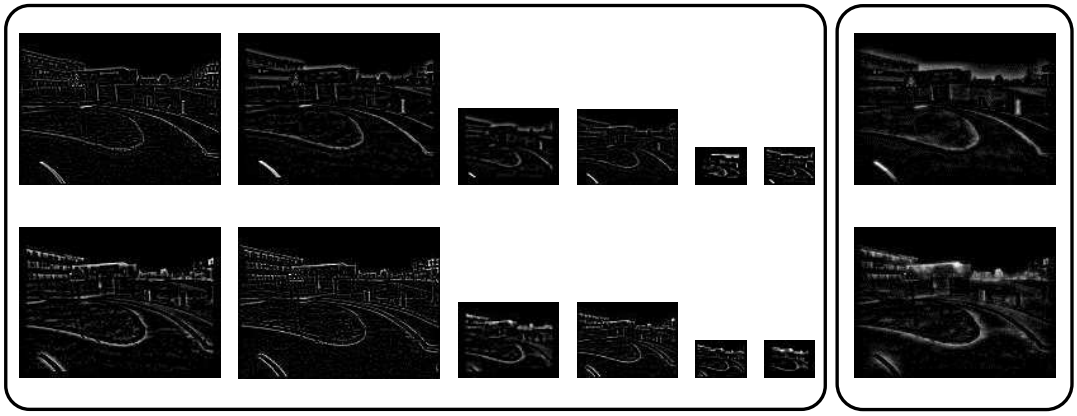
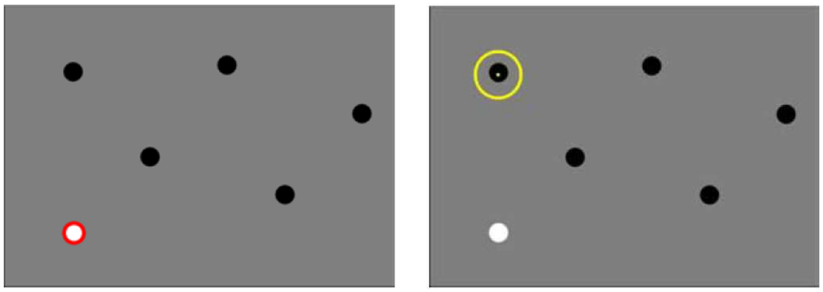


FIGURE 2.9: À Gauche : 12 cartes d'intensité à différentes échelle. Première rangée : « *on-center* ». Deuxième rangée : « *off-center* ». À droite : les deux cartes caractéristiques d'intensité I_{on} et I_{off} résultantes de la sommation des cartes d'intensités sur la gauche.

Les calculs dans le modèle de Frintrop (2006) [56] diffèrent de ceux de Itti et al. (1998) [84] car dans le premier on calcule le « *on-center* » et « *off-center* » séparément, tandis que dans le second, ces calculs sont combinés en prenant la valeur absolue de la différence |*Centre-Contour*| directement. Cette approche est une approximation plus rapide du système visuel humain, mais génère quelques problèmes. Tout d'abord, un phénomène de pop-out correct n'est pas garanti : imaginez une image avec un fond gris et des points blancs et noirs sur elle, les deux ayant le même contraste d'intensité à l'arrière-plan (Figure 2.10). S'il y a un seul point blanc, mais plusieurs noirs, le blanc sautera aux yeux de l'observateur humain. Une condition pour ce pop-out est la séparation des mécanismes « *on-center* » et « *off-center* » : le premier mécanisme répond aux points blancs tandis que le second aux noirs. La combinaison des deux dans une carte

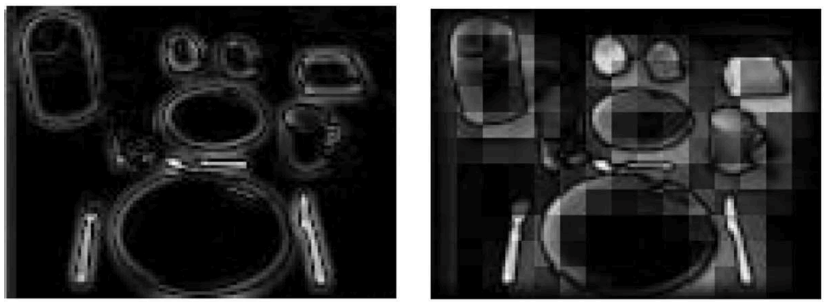
comme l'a fait Itti et al. (1998) [84] ne permet pas le pop-out. Les résultats de la détection des deux modèles sur l'exemple de pop-out introduit ci-dessus sont représentés sur la Figure 2.10.



(a) Premier FOA avec VOCUS. (b) Premier FOA avec NVT.

FIGURE 2.10: Le point blanc est détecté par VOCUS mais pas par le NVT.

Les deux modèles varient également dans le calcul des différences elles-mêmes. Dans le modèle présenté par Itti et al. (1998) [84], les différences sont déterminées par la soustraction de deux échelles à la fois, par exemple, $I_6 = s_4 - s_8$. Le problème avec cette approche est qu'elle donne des sortes de cartes ayant un texturage carrée et des discontinuités aux bords à l'échelle grossière (voir Figure 2.11 à droite). Ce problème est corrigé en utilisant l'approche de Frintrop (2006) [56] (voir Figure 2.11 à gauche), avec un calcul un peu plus lent, mais beaucoup plus précis et qui a besoins de moin d'échelles.



(a) Carte d'intensité obtenu avec VOCUS. (b) Carte d'intensité obtenu avec NVT.

FIGURE 2.11: Deux cartes d'intensité d'une scène de la table du petit déjeuner, calculées par VOCUS et par NVT. La structure carrée résultant de la différence entre deux échelles peut être vue clairement dans l'image de droite, l'image de gauche montre une solution beaucoup plus fine.

Pour le canal de couleur, l'image en couleur est d'abord convertie en une image dans l'espace CIE-Lab[2]. L'espace de couleur CIE-Lab est actuellement l'un des espaces de couleur uniforme les plus populaires [53]. L'utilisation de cet espace a abouti à des améliorations considérables aux niveaux des résultats par rapport à d'autres [56].

De l'image Lab, une pyramide d'image P_{Lab} est générée par l'application d'un filtre Gaussien sur l'image Lab. De P_{Lab}, quatre pyramides de couleurs P_R, P_G, P_B et P_Y sont générées pour les couleurs distinctes rouge, vert, bleu et jaune. Notez que P_{Lab}

2 CIE : Commission Internationale de l'Éclairage, « L » représente la luminosité, « a » représente l'axe de rouge/vert, et « b » représente l'axe bleu/jaune

est une pyramide d'images en couleurs, ainsi un pixel d'une couche dans la pyramide est un vecteur (P_L, P_a, P_b), alors que P_R, P_G, P_B et P_Y sont des pyramides d'images avec une seule composante, ainsi un pixel d'une couche de ces pyramides est un scalaire.

La valeur d'un pixel $P_{(R,s)}(x,y)$ du niveau « s » de la pyramide « rouge » P_R est obtenue par la distance entre le pixel $P_{Lab}(x,y)$ et le prototype de la couleur rouge « $r = (r_a, r_b) = (255, 127)$ » pour une valeur maximale de 255 dans l'espace de couleur. Comme $P_{Lab}(x,y)$ est de la forme (P_a, P_b), cela donne :

$$\begin{aligned} P_{(R,s)}(x,y) &= d(P_{Lab}(x,y), r) \\ &= \| P_{Lab}(x,y) - r \| \end{aligned} \tag{2.2}$$

où « $d(a,b)$ » est la distance entre a et b.

La représentation de rouge et vert ainsi que de bleu et de jaune avec des cartes distinctes permet de trouver les éléments saillants suivant une couleur spécifique, et faire aussi une recherche Top-Down d'une couleur à la fois. En revanche, dans les calculs combinés rouge-vert et bleu-jaune comme dans le modèle NVT [84] certaines couleurs ne peuvent pas être saillantes et une recherche de couleur spécifique n'est pas possible.

Les cartes de ces pyramides de couleurs montrent à quel degré une couleur est présente dans une image, exemple : les cartes de P_R reflètent à quel point la couleur rouge est présente dans l'image : les valeurs les plus élevées correspondent à des régions rouges et les valeurs les plus faibles aux régions vertes (car c'est le vert qui a la plus grande distance au rouge dans l'espace de couleur).

Les cartes de couleurs pour la table de baby-foot sont présentées dans la Figure 2.12 (première rangée). Sur ces pyramides, le contraste de couleur est calculé en utilisant la partie « *on-center* » de l'algorithme de *Centre-Contour* décrit dans l'Algorithme 1, ce qui donne au final $4 \times 3 \times 2 = 24$ cartes :

$$C_{(\gamma,s,\sigma)} = \mathrm{Centre} - \mathrm{Contour}_{(on,s,\sigma)} \tag{2.3}$$

Avec : $\gamma \in \{r, g, b, y\}$, $s \in \{s_2, s_3, s_4\}$ et $\sigma \in \{3, 7\}$

La partie « *off-center* » de l'algorithme *Centre-Contour* n'est pas nécessaire, parce que ces valeurs sont représentées dans la pyramide de la couleur opposée à celle traitée. Les cartes de chaque couleur sont réechelonnées au niveau s_2 et sommées en 4 cartes de caractéristique de type couleur :

$$C_\gamma = \bigoplus_{s,\sigma} C_{(\gamma,s,\sigma)} \tag{2.4}$$

Avec : $\gamma \in \{r, g, b, y\}$, $s \in \{s_2, s_3, s_4\}$ et $\sigma \in \{3, 7\}$

Les cartes de caractéristique (couleur) de l'image de la table de baby-foot sont présentées dans la Figure 2.12 (deuxième rangée).

Pour le canal orientation, les cartes de caractéristiques pour l'orientation sont obtenues à partir de la convolution de la pyramide gaussienne issue de la composante intensité I par la pyramide de Gabor orientée $O_{(\theta,s)}$ où θ représente l'orientation choisi et s le niveau dans la pyramide [64]. Dans ce modèle ainsi que dans les précédents, quatre orientations sont considérées $\theta \in \{0°, 45°, 90°, 135°\}$.

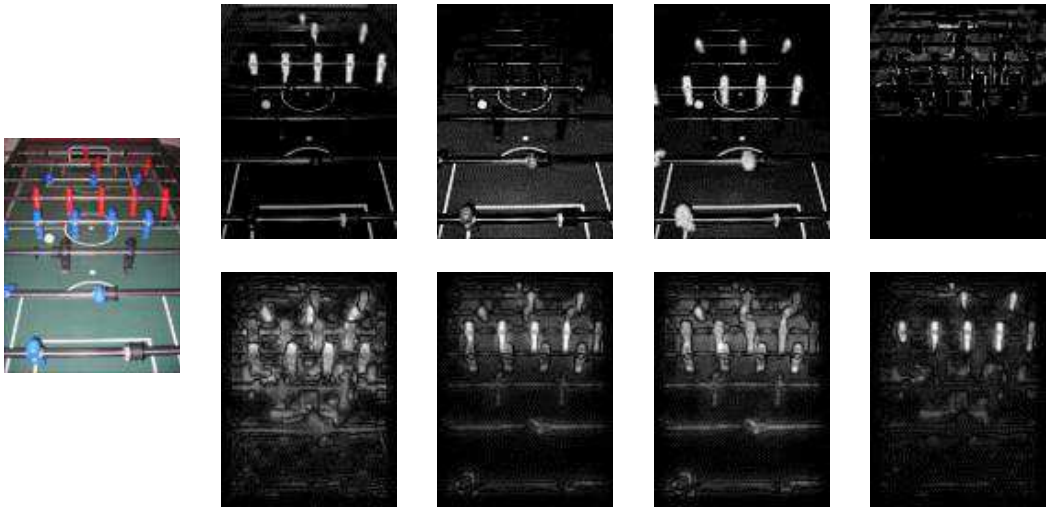
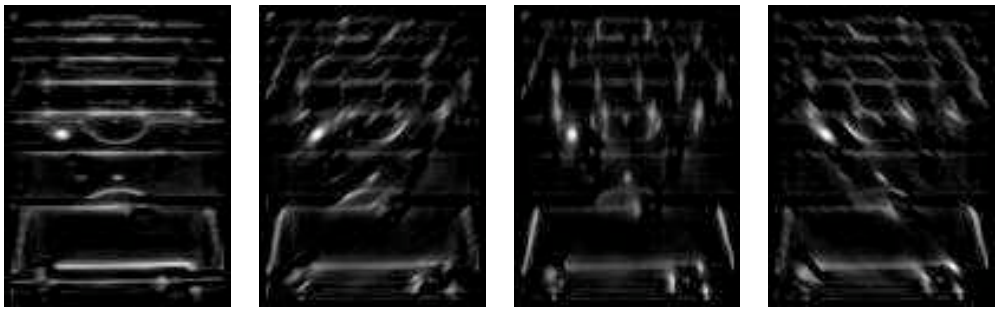


FIGURE 2.12: À gauche : l'image d'entrée. À Droite : première rangée : les cartes du niveau s_2 des pyramides de couleurs P_R, P_G, P_B et P_Y (rouge, vert, bleu et jaune). Deuxième rangée : les cartes de caractéristique de couleur qui en résultent après l'application de l'algorithme *Centre-Contour*.

Contrairement au modèle NVT de Itti et al. (1998) [84], le modèle de Frintrop (2006) [56] n'utilise pas la technique *Centre-Contour* explicitement pour le calcul des cartes caractéristiques d'orientation. La différence *Centre-Contour* orientée qui est déterminée par des cellules dans le cortex humain est déjà déterminée implicitement par les filtres de Gabor. Donc, les cartes d'orientations sont prises directement, ce qui donne $3 \times 4 = 12$ cartes d'orientations $O_{(\theta,s)}$, pour les orientations $\theta \in \{0°, 45°, 90°, 135°\}$ et les échelles $s \in \{s_2, s_3, s_4\}$. Les cartes d'orientation $O_{(\theta,s)}$ sont additionnées suivant chaque orientation, ce qui donne quatre cartes caractéristiques d'orientation O_θ correspondant à l'échelle du niveau s_2, une pour chaque orientation : avec $\theta \in \{0°, 45°, 90°, 135°\}$ et $s \in \{s_2, s_3, s_4\}$. Les cartes caractéristiques d'orientation sont représentées sur la Figure 2.13.



(a) $O_{0°}$ (b) $O_{45°}$ (c) $O_{90°}$ (d) $O_{135°}$

FIGURE 2.13: Les quatre cartes caractéristiques d'orientation.

Après le calcul de toutes les cartes de caractéristiques, la carte de saillance est obtenue en combinant ces différentes caractéristiques. Une des difficultés pour cette combinaison est que ces cartes représentent des données a priori non comparables à des échelles différentes. Des objets saillants dans seulement quelques cartes peuvent

être masqués par le bruit ou par d'autres objets moins saillants des autres cartes. Pour pallier à cette difficulté une étape de *normalisation* et de *pondération* sont nécessaire.

La fusion des différentes cartes est effectuée par une moyenne pondérée : les cartes sont d'abord pondérées par une fonction de pondération, puis elles sont additionnées, et enfin normalisées.

Si les cartes étaient additionnées de façon simple, toutes les cartes auraient la même influence. Cela implique que si il y a beaucoup de cartes, l'influence de chaque carte est très faible et ses valeurs ne contribuent pas beaucoup à la carte finale de saillance qui en résulte. Pour éviter cet effet, la première étape consiste à déterminer les cartes les plus importantes et augmenter leur influence. Ceci peut être réalisé par un opérateur comme l'opérateur de pondération présenté dans [84] pour la carte X :

$$N(X) = X \times (M - \overline{m})^2 \tag{2.5}$$

où M est le maximum global et $\overline{m}$ est la moyenne des maximas locaux. Cette technique permet de donner plus d'importance aux cartes avec un pic fort et supprimer ceux qui contiennent de nombreux sommets presque équivalents Figure 2.4.

Le problème de cette approche décrit dans l'article de Itti and Koch (2001) [83], est qu'elle donne un bon résultat lorsqu'il y a juste un seul point saillant, car si il y a deux maxima seulement, la différence donne zéro, ignorant la carte complètement, alors que le système visuel humain considéra les deux comme des points saillants. Le même article propose un schéma itératif complexe et computationnellement coûteux pour résoudre ce problème par la concurrence locale entre les endroits saillants voisins. Frintrop (2006) [56] dans ses travaux a proposé une approche alternative plus simple en divisant chaque carte par la racine carrée du nombre de maxima locaux présent dans la carte :

$$W(X) = \frac{X}{\sqrt{m}} \tag{2.6}$$

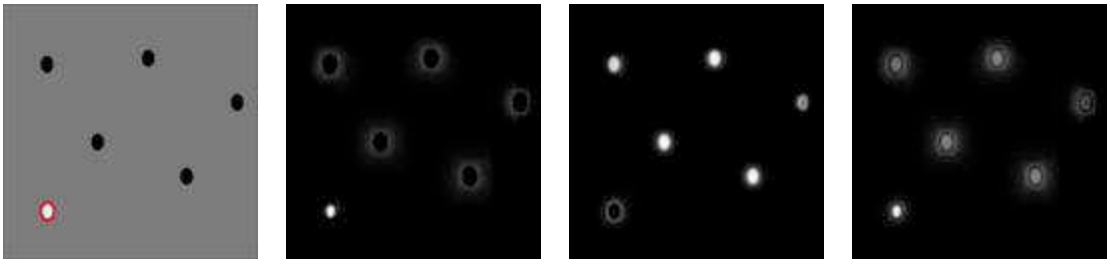
Avec : X représentant la carte à normaliser et m le nombre de maxima locaux supérieur à un seuil, c'est-à-dire le nombre de régions saillantes extraites de la carte. D'après les expériences et travaux de Frintrop (2006) [56] la méthode la plus simple est de considérer un seuil de 50% du maximum global. La Figure 2.14 donne un exemple d'utilisation de cette approche : la carte I_{on} avec un seul pic et la carte I_{off} avec cinq pics.

L'étape suivante dans le calcul de saillance est la génération des cartes d'évidence. Pour obtenir ces cartes, toutes les cartes d'une caractéristique sont pondérées par la fonction de pondération défini précédemment *W* et combinées en une seule carte d'évidence, ce qui donne la carte I pour l'intensité, O pour l'orientation et C pour la couleur :

$$\begin{cases} I & = \sum_i W(I_i) \quad \text{avec} \quad i \in \{on, off\} \\ O & = \sum_\theta W(O_\theta) \quad \text{avec} \quad \theta \in \{0°, 45°, 90°, 135°\} \\ C & = \sum_\gamma W(C_\gamma) \quad \text{avec} \quad \gamma \in \{rouge, vert, blue, jaune\} \end{cases} \tag{2.7}$$

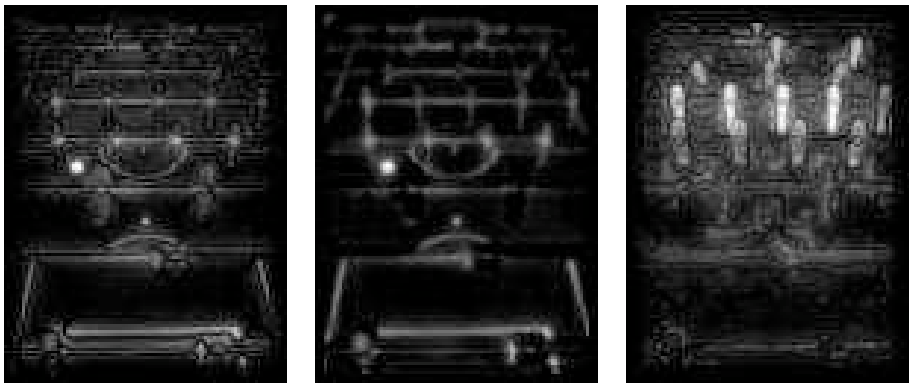
La Figure 2.15 donne un exemple des trois cartes d'évidences obtenues lors de traitement de l'image (baby-foot).

Après avoir sommé les cartes de caractéristiques pondérées, une normalisation doit être faite pour rendre les cartes d'évidence comparables. Cela est nécessaire car certaines



(a) I (b) I_{on} (c) I_{off} (d) I_S

FIGURE 2.14: L'effet de la fonction de pondération $W(X)$ (a) l'image d'entrée, (b) carte caractéristique d'intensité on-center, (c) carte caractéristique d'intensité off-center, (d) la carte d'évidence d'intensité. I_{on} a un poids plus élevé que I_{off}, car il ne contient qu'un seul pic intense donc cette carte à une plus grande influence et le point blanc apparaît dans la carte d'évidence.



(a) I (b) O (c) C

FIGURE 2.15: Les cartes d'évidences : d'intensité (a), d'orientation (b), et de la couleur (c).

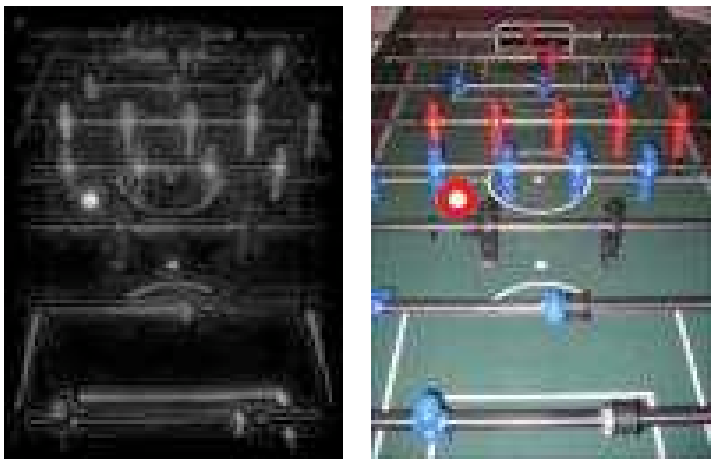
caractéristiques ont plus de cartes que d'autres. La fonction $N(\square)$ proposée par Itti et al. (1998) [84] normalise les cartes sur une plage fixe. Le problème de cette méthode est que la normalisation des cartes sur une plage fixe supprimera des informations importantes sur l'amplitude de certaines cartes.

Une autre approche qui a donnée de meilleurs résultats est celle proposé par Frintrop (2006) [56] où chaque carte d'évidence doit être normalisée séparément, exemple pour la caractéristique d'intensité :

$$\begin{cases} \text{Trouver le maximum des cartes } I_i & \widehat{m}_I = max(\acute{I}_i) \\ \text{Additionner les cartes } I_i & I = \sum_i W(\acute{I}_i) \\ \text{Normaliser } I & I = n(0, \widehat{m}_I)(I) \end{cases} \tag{2.8}$$

Enfin, les cartes d'évidence I, O et C sont pondérées de nouveau avec l'approche déjà présentée avec l'utilisation de $W(\square)$ et additionnées pour aboutir à la carte de saillance finale S :

$$S = W(I) + W(O) + W(C) \tag{2.9}$$



(a) Carte de saillance S (b) FOA

FIGURE 2.16: Pour obtenir le centre de l'attention FOA, la région la plus saillante MSR est déterminée de la carte de saillance S.

La carte de saillance pour l'exemple de la table « baby-foot » est illustrée dans la Figure 2.16a. La carte de saillance finale donne les points saillants de l'image, chacun ayant une activité plus ou moins forte selon l'importance du point. Elle se calcule en combinant les cartes d'évidences obtenues précédemment par une sommation naïve :

$$S = \frac{1}{3} \times (I + O + C) \tag{2.10}$$

ou pondérée :

$$S = w_I \times I + w_O \times O + w_C \times C \tag{2.11}$$

A partir de cette carte de saillance, les points d'attention visuelle peuvent être extraits. Le pixel de valeur maximum défini le point le plus saillant de l'image, celui vers lequel l'attention du regard se porte. Comme avec la vision humaine, le processus d'IOR est utilisé pour permettre au système de trouver le prochain point le plus saillant et ainsi de suite. Le résultat final est la liste des coordonnées des points saillants.

2.3.3 *D'autres modèles psycho-visuels*

Le modèle de Milanese (1993) [127], [128], suit l'architecture de Koch and Ullman (1987) [98]. D'abord des caractéristiques visuelles telles que intensité, couleur et contours sont extraites de l'image, puis filtrées par un filtre passe bande orienté. L'intérêt de ce modèle réside dans le mécanisme de fusion des cartes de saillance de chaque canal pour aboutir à la carte de saillance finale.

Le modèle de Chauvin et al. (2002) [33], [34] suit quant à lui le modèle de Itti et al. (1998) [84] et se différencie de l'implantation d'origine par quelques étapes comme par exemple l'extraction des primitives visuelles pré-attentives qui s'effectue à partir d'une base de 32 ondelettes de Gabor couvrant 4 bandes de fréquences et 8 orientations.

Le modèle de Bruce and Jernigan (2003) [25] inspiré du modèle de Itti et al. (1998) [84] a la particularité d'intégrer dans une architecture classique de type Koch and Ullman (1987) [98] un opérateur statistique basé sur le même principe que celui utilisé par Oliva et al. (2003) [149]. Ce principe, énoncé auparavant, lie la saillance d'un évènement à sa probabilité d'apparition.

2.4 APPLICATIONS

Ce chapitre, se focalise plus sur la description des modèles de l'attention eux-mêmes. Il y a, cependant, de nombreuses applications technologiques de ces modèles qui ont été développées au fil des ans et qui ont encore augmenté l'intérêt de la modélisation de l'attention. Ces applications sont organisées en trois catégories : vision et graphique, robotique, autres domaines, dans le Tableau 2.1.

2.5 CONCLUSION

L'attention visuelle est un mécanisme essentiel pour le système visuel humain, car elle permet la sélection des parties visuellement pertinentes de la scène. En outre, l'attention visuelle explique partiellement la forte performance de notre système de vision.

Selon le modèle de l'attention de Koch and Ullman (1987) [98], l'étude c'est concentrée sur les trois caractéristiques mentionnées dans ce chapitre à savoir l'intensité, l'orientation et la couleur, car ils appartiennent à des éléments de base de la perception humaine et sont assez faciles à modéliser.

Considérant que plus de caractéristiques améliore la qualité mais ralentit le système car le traitement parallèle est réalisé en série sur les machines actuelles. Néanmoins, l'architecture du système permet une extension facile à plus de caractéristiques et aussi une éventuelle implémentation du modèle sur des circuits dédiés favorisant le traitement en parallèle comme par exemple les FPGA(s).

CATÉGORIE	APPLICATIONS	RÉFÉRENCES
Vision par Ordinateur et Graphismes	Segmentation d'image	Mishra and Aloimonos (2009) [129], Maki et al. (2000) [118]
	L'appariement d'images	Walther and Koch (2006) [189], Siagian and Itti (2009) [170], Frintrop and Jensfelt (2008) [58]
	Compression d'images	Itti (2004) [81], Guo and Zhang (2010) [65]
	Classification de scène	Siagian and Itti (2007) [169]
	Détection d'objet	Frintrop (2006) [56], Navalpakkam and Itti (2006) [140], Fritz et al. (2005) [59], Butko and Movellan (2009) [30], Viola and Jones (2004) [188]
	Détection d'objet saillant	Liu et al. (2011) [114], Goferman et al. (2012) [62], Achanta et al. (2009) [3], Rosin (2009) [162]
	Reconnaissance d'objets	Salah et al. (2002) [164], Walther and Koch (2006) [189], Frintrop (2006) [56], Mitri et al. (2005) [130], Gao and Vasconcelos (2004) [61], Han and Vasconcelos (2010) [67], Paletta et al. (2005) [152]
	Suivi visuel	Mahadevan and Vasconcelos (2009) [117], Frintrop (2010) [57]
	L'éclairage dynamique	El-Nasr et al. (2009) [50]
	Détection de points d'intérêt	Kadir and Brady (2001) [90]
	Segmentation et suivi du visage	Li et al. (2008) [112]
Robotiques	Vision active	Borji et al. (2010) [21], Dankers et al. (2007) [43],Vijayakumar et al. (2001) [187], Mertsching et al. (1999) [122]
	Localisation de robot	Ouerhani (2004) [151], Siagian and Itti (2009) [170]
	Navigation de robot	Scheier and Egner (1997) [167], Baluja and Pomerleau (1997) [12]
	Interaction homme-robot	Breazeal and Scassellati (1999) [24], Heidemann et al. (2003) [70], Nagai (2009) [136], Muhl et al. (2007) [134]
	Vision synthétique	Courty and Marchand (2003) [39]
Autres	Publicité	Rosenholtz et al., Liu et al. (2011, 2008) [161, 113]
	Trouver des tumeurs dans les mammographies	Hong and Brady (2003) [76]
	Prothèse rétinienne	Parikh et al. (2010) [153]

TABLE 2.1: Quelques applications de la modélisation de l'attention visuelle [181].

3

NAVIGATION PAR SAILLANCE : SAILMAP

3.1 INTRODUCTION

La cartographie visuelle est devenue un élément fondamental de la robotique mobile. Lorsque l'on considère des environnements à cartographier de grandes tailles les approches topologiques sont très efficaces. En effet, les cartes topologiques facilitent la représentation et la gestion des données de capteurs en raison de leur indépendance explicite vis à vis d'information métrique.

La détection de fermeture de boucle, qui consiste en la reconnaissance des lieux précédemment explorés, est en effet nécessaire et un processus très important dans la construction de carte précise et non redondante. De nombreuses solutions ont été proposées dans la littérature pour résoudre ce problème de manière efficace [41], [42], [6] et [60]. Lors de la navigation le robot peut passer par le même endroit à plusieurs reprises, mais ne jamais observer l'environnement dans les mêmes exactes conditions (luminosité, pose, objets mobiles, ...). C'est pour cela qu'un bon détecteur de fermeture de boucle doit tenir compte de la variabilité inhérente de ces paramètres.

La première étape vers l'élaboration d'un détecteur de fermeture de boucle est le choix entre une méthode en ligne ou bien hors ligne. Les deux sont abordées dans la littérature. En général le choix dépend de l'objectif à atteindre et/ou de l'application. Par exemple, une méthode hors ligne est suffisante si le robot a déjà visité tout l'environnement à cartographier. Quant aux méthodes en ligne (temps réel), elles sont nécessaires lorsque la cartographie de l'environnement et la localisation du robot sont réalisées en même temps (SLAM).

Lorsqu'un système de vision est utilisé pour la détection de fermeture de boucle, il s'agit alors de mesurer des vraisemblances entre les images, en exploitant des primitives locales ou globales. Les techniques exploitant les primitives globales sont sensibles aux grandes distorsions, occultations et changements induit par le mouvement, l'apparition ou la disparition d'objet dans la scène. Tandis que les primitives locales ont une grande précision et peuvent gérer les petits changements qui apparaissent dans l'image [202]. Par contre un descripteur est nécessaire pour pouvoir distinguer les différentes régions entre elles. Par exemple, un descripteur simple serait un vecteur constitué par les valeurs d'intensité des pixels entourant le point saillant détecté. Toutefois, ces primitives ne sont pas directement discriminantes. Elles ne se distinguent qu'en combinant leurs descripteurs locaux avec leur emplacement absolu ou relatif dans l'image [75]. Différents détecteurs et descripteurs de primitives ont été développés au cours des années précédentes. Une évaluation des descripteurs les plus puissants peut être trouvée dans [126]. Dans [41], [42] et [6] les descripteurs utilisés sont SIFT et SURF. Ce sont les plus populaires dans ce contexte, car ils sont invariants à l'échelle, à la rotation ainsi qu'au changement d'éclairage, et se comportent correctement pour de légers changements de perspective.

Parmi ces primitives, les Régions Saillantes (Region Saillante (RS)) sont connus pour être très discriminantes, robustes aux changements de point de vue, aux mauvaises conditions météorologiques (tels que le brouillard ou la pluie) et aux perturbations d'image, ainsi que pour avoir un taux de répétabilité élevé (sur les 5 à 20 premières régions détectées sur une image) [57]. En plus des avantages précédemment cités des RS, il faut mentionner que ces primitives ne se basent pas sur des formes ou des caractéristiques spécifiques trouvées dans l'environnement (comme la verticalité dans des environnements urbains, ou la planéité locale dans la plupart des environnements structurés). Au final, cela signifie que notre approche basée sur ce type de primitives qui est nommé **SAILMAP** (pour SAILENCY MAPPING) n'est pas limitée à un type particulier d'environnement. En outre, le temps de calcul ainsi que les exigences de mémoire sont réduites, en raison du nombre limité de régions nécessaires par image.

Dans ce chapitre nous présentons une nouvelle méthode basée sur la vision pour la détection de fermeture de boucle et une stratégie pour résoudre le problème de localisation d'un robot dans une carte topologique dense. Deux méthodes d'appariement d'images sont présentées : *SAILMAP* et *Prob-SAILMAP* qui se base sur un filtre Bayésien récursif. L'une des principales nouveautés scientifiques de cette étude réside dans le fait que dans la seconde approche *Prob-SAILMAP* la mémoire visuelle est construite et maintenue au niveau des primitives. Au lieu de stocker des images clés, la mémoire stocke directement des primitives, qui peuvent être associées à un ensemble arbitraire d'images. La Figure 3.1 donne un aperçu des étapes de chaque approche. Le bloc de détection et de description de primitives saillantes génère pour chaque image F_n traité le vecteur suivant :

$$V_{F_n} = \{x_k, y_k, V_{SR_k}\}_{k \in 1,m} = \{D_k\}_{k \in 1,m} \tag{3.1}$$

où : m représente le nombre de régions saillantes détectées, (x_k, y_k) et V_{SR_k} représentent les coordonnées et le descripteur (voir section 3.3) de chaque région dans l'image F_n, respectivement. Plus de détails concernant les étapes de chaque approche seront données dans les sections suivantes.

Dans la section suivante nous présentons les travaux trouvés dans la littérature, et qui traitent le problème de détection de fermeture de boucle. Dans la section 3.3, nous présentons le descripteur que nous avons choisi d'utiliser dans nos différents travaux. La section 3.4 introduit la représentation d'environnement par carte topologique. Dans cette section, plus de détails sont données sur la construction de la carte topologique (mémoire visuelle de l'environnement) dans l'approche *SAILMAP*. Les méthodes utilisées pour la recherche d'image par similarité dans les approches *SAILMAP* et *Prob-SAILMAP* sont présentées à la section 3.5. Les différents résultats sont données dans la section 3.7.

3.2 TRAVAUX CONNEXES ET ARCHITECTURE GÉNÉRALE

La plupart des techniques trouvées dans la littérature se différencient par le procédé de détection de fermeture de boucle. Généralement, ces techniques ont en commun, pour fonctionner de façon efficace, l'utilisation d'un modèle de sac-de-mots (Bag of Words (BoW)) [41], [42], [60] et [55], et l'utilisation des fichiers inversés [6]. La représentation par sac-de-mots est une description de document très utilisée en recherche d'information, originaire du traitement du langage naturel, mais qui a été transposée dans le domaine

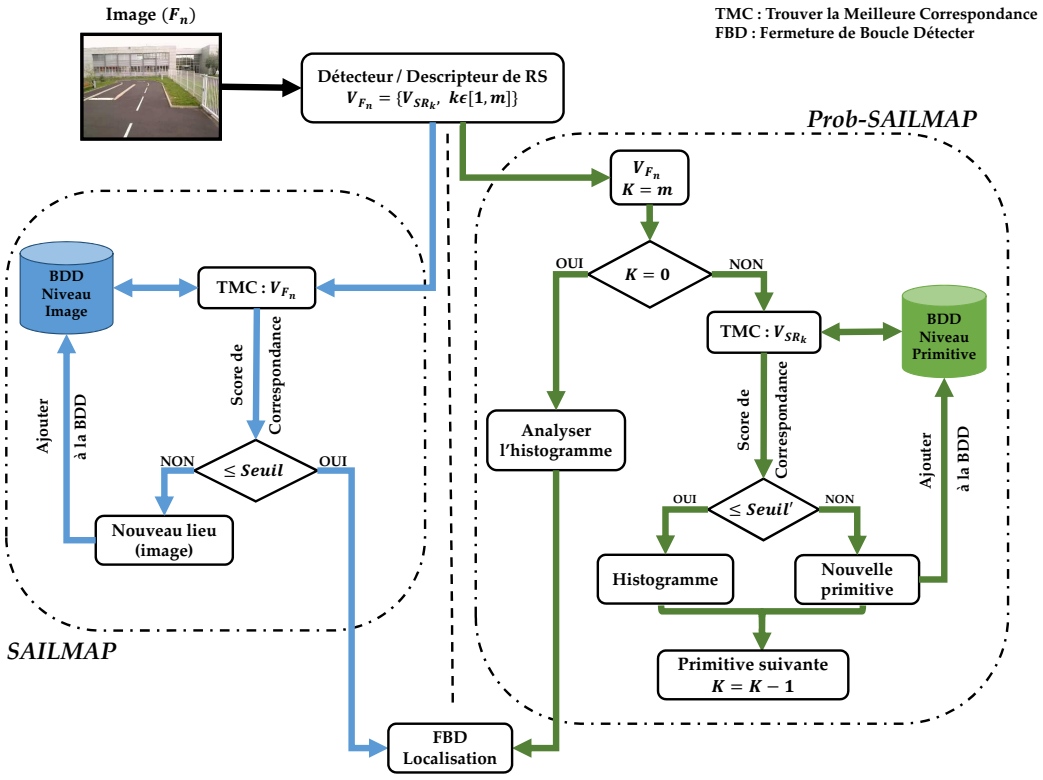


FIGURE 3.1: Les différentes étapes de l'approche *SAILMAP* (à gauche) et *Prob-SAILMAP* (à droite).

de la vision par ordinateur, où les documents (lieux ou images) sont représentés par un ensemble de mots visuels (primitives).

Plus précisément, les mots visuels sont définis par un modèle génératif dans [41] et [42]. Des histogrammes de mots sont utilisés pour la détection de fermeture de boucle dans [55]. Une autre approche semblable utilisant le descripteur BRIEF est présenté dans [60]. L'approche présentée dans [41] et [42] connu sous le nom de FABMAP, détecte les fermetures de boucle dans une séquence d'images acquisent avec une caméra omnidirectionnelle et atteint une *Précision*[1] de 100% sur certains circuits, mais avec un *Rappel*[2] de 48% et seulement de 3% pour des trajectoires de 70km et 1000km respectivement. Ces deux paramètres seront utilisés par la suite dans la section 3.7 pour évaluer notre approche et la comparer à FABMAP.

L'approche FABMAP est devenue la référence pour la détection de fermeture de boucle, mais sa robustesse diminue lorsque de nombreuses images contiennent des structures très similaires. En effet, cette approche repose sur un ensemble non-structuré de mots visuels et par conséquent, ne reflète pas l'information géométrique codée des caractéristiques extraites. Bien qu'elle permette une adaptation très efficace, elle souffre d'aliasing perceptuel, car les caractéristiques extraites sont arbitrairement réorganisées et contraintes à être égales à un des mots visuels [168].

1 La *Précision* (%) est calculée comme le pourcentage de fermetures de boucle détectées qui sont correctes.
2 Le *Rappel* (%) est calculée comme le pourcentage de fermetures de boucle détectées correctes par rapport à la réalité terrain.

Dans l'approche FABMAP, ainsi que dans *SAILMAP* présenté dans ce chapitre, les cartes construites sont des cartes topologiques denses, où chaque image est considérée et est représentée par un nœud.

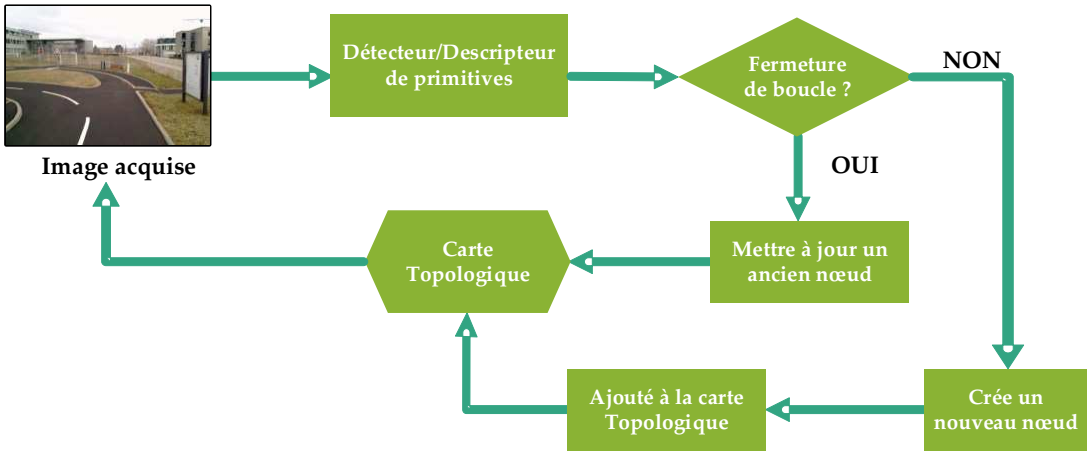


FIGURE 3.2: Une vue globale modulaire de l'approche utilisée pour cartographier topologiquement un environnement.

La Figure 3.2 présente une vue d'ensemble générale du cadre de travail présenté dans ce chapitre, pour la construction de carte topologique basée sur la vision. Étant donnée une image de la séquence ou une image nouvellement acquise, les caractéristiques locales de l'image de type saillance sont extraites à l'aide d'un détecteur et d'un descripteur dédiés. Ces caractéristiques extraites sont utilisées pour détecter les éventuelles fermetures de boucle, et le module évalue pour cela la similitude entre l'image courante et toutes les images associées aux nœuds de la carte topologique. Afin d'optimiser cette détection de nombreuses optimisations sont envisageables comme la recherche par partitionnement de l'espace des descripteurs globaux d'images et/ou la recherche par priorité de lieux, en testant d'abord les nœuds connectés au nœud courant.

Quand une fermeture de boucle est détectée, cela signifie que l'image requête a probablement été acquise dans un lieu/nœud précédemment visité. Si aucune fermeture de boucle n'est trouvée, un nouveau lieu/nœud est créé et ajouté à la carte en le connectant au nœud précédemment exploré.

3.3 DESCRIPTEUR

L'algorithme de détection de RS génère une primitive à la fois, dans l'ordre décroissant de saillance, et les premières (< 20) sont généralement assez discriminantes et hautement reproductibles. Pour la détection et la description de primitives, nous utilisons une version modifiée de l'algorithme original de Frintrop (2006) [56], qui a produit des résultats significatifs dans le domaine de la robotique mobile.

Pour chaque RS, un vecteur de caractéristique V_{RS} composé de 10 éléments est défini. Ce vecteur représente le niveau de contribution de chaque caractéristique à la carte associée. Les trois cartes d'évidence considérées sont l'intensité (I), la couleur (C) et

l'orientation (O). Ces cartes s'obtiennent à partir des cartes de caractéristiques suivante : I_{on} et I_{off} pour l'intensité, C_R, C_V, C_B et C_J pour la couleur, et O_0, O_{45}, O_{90} et O_{135} pour l'orientation (voir chapitre 2). Chaque élément du vecteur V_{RS} est défini comme étant le rapport de la valeur moyenne de la RS dans la carte de caractéristique X_i (noté $\overline{X_i(SR)}$) sur la valeur moyenne de la même région dans la carte d'évidence CX_i lui correspondant.

$$V_{SR} = \{v_i\}_{i \in 1,10} \tag{3.2}$$

Où :

$$v_i = \frac{\overline{X_i(SR)}}{\overline{CX_i(SR)}} \tag{3.3}$$

Ce descripteur a été inspiré par celui proposé par Frintrop (2006) [56], qui est composé de 13 éléments obtenus à partir des 10 cartes de caractéristiques et 3 cartes d'évidences du modèle VOCUS [56]. La valeur de chaque élément v_i est défini comme étant le rapport de la moyenne de la région saillante sur la moyenne du reste de la carte X_i ($\neg$RS), c'est-à-dire, l'arrière plan de la RS :

$$v_{i_{Frintrop}} = \frac{\overline{X_i(SR)}}{\overline{X_i(\neg SR)}} \tag{3.4}$$

Ce descripteur a été utilisé principalement pour des applications de suivi où les régions à apparier appartiennent à des images successifs, dans ce contexte l'arrière-plan ne change pas de manière significative. Dans le cas de cette étude, qui a comme objectif l'utilisation des RS pour la détection de fermeture de boucle et la localisation, les images à mettre en correspondance ne sont pas acquises successivement ce qui peut conduire à des changements considérables entre deux images (point de prise de vue différente, ciel, présence ou absence d'objet, . . .), comme illustré sur la Figure 3.3. Ces changements auront un impact considérable sur le descripteur proposé par Frintrop (2006) [56].

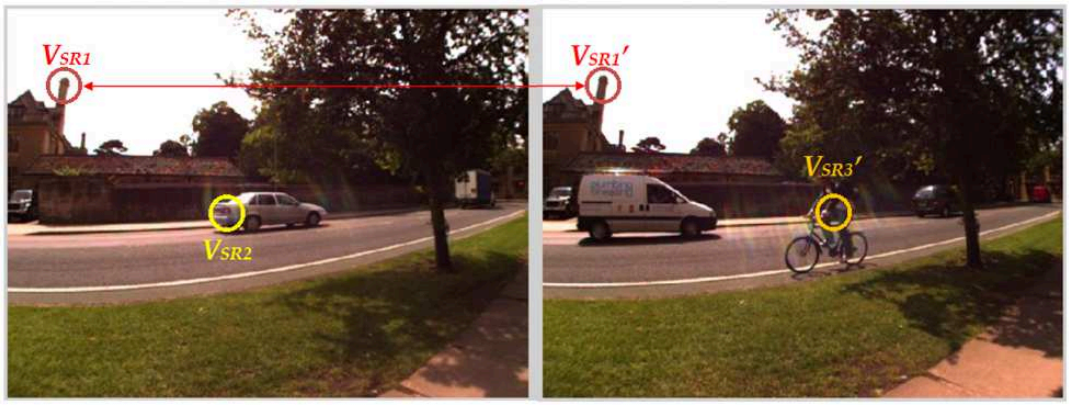


FIGURE 3.3: Deux images issues des ensembles de données de l'Université d'Oxford prisent au même endroit. Changements importants : présence d'objets différents (voiture, vélo).

Le Tableau 3.1 donne les différentes valeurs obtenu avec notre descripteur et celui Frintrop (2006) [56], pour la régions saillantes V_{SR_1} et V'_{SR_1} (voir la Figure 3.3). On remarque bien qu'avec notre descripteur la différence euclidienne (de 0.4778) qu'il y'a entre V_{SR_1} et $V_{SR'_1}$, représentant le même objet saillant, est beaucoup plus faible que la distance (de 6.4349) obtenu avec le descripteur de Frintrop (2006) [56] .

NOTRE DESCRIPTEUR		LE DESCRIPTEUR DE FRINTROP	
V_{SR_1}	$V_{SR'_1}$	V_{SR_1}	$V_{SR'_1}$
2.4370	2.3315	9.3351	8.4026
0.9225	0.7881	3.5758	2.9148
1.2626	1.1937	6.4025	5.7964
0.5697	0.5571	2.9170	2.7677
1.0912	1.0664	5.4984	5.0972
0.4819	0.4585	2.4325	2.1660
1.1013	0.8173	3.3579	2.4507
1.2398	0.8875	3.7823	2.6574
1.0412	0.8308	3.1618	2.4731
1.4041	0.9642	4.2765	2.8822
$\lvert V_{SR_1} - V_{SR'_1}\rvert^2$	0.4778	$\lvert V_{SR_1} - V_{SR'_1}\rvert^2$	6.4349

TABLE 3.1: Comparaison de notre descripteur avec celui de Frintrop.

Lors de la détection des régions saillantes sur une image F_n, les coordonnées de chaque RS : (x_k, y_k) sont également conservées. Celles-ci seront utilisées pour imposer des contraintes géométriques et ainsi améliorer les résultats de mise en correspondance. Pour chaque image F_n de laquelle m RS différentes sont extraites, un descripteur global d'image est défini comme suit :

$$V_{F_n} = \{x_k, y_k, V_{SR_k}\}_{k\in 1,m} = \{D_k\}_{k\in 1,m} \tag{3.5}$$

Les composants de V_{F_n} sont classés par ordre décroissant de saillance. Cela permettra d'obtenir dans l'approche *SAILMAP* une carte topologique de l'environnement dense où chaque image sera représentée par un nœud contenant les m primitives (RS) détectées dans l'image. Les descripteurs d'image peuvent être par la suite quantifiés pour réduire les besoins en mémoire et le temps de calcul des processus de détection de fermeture de boucle ou de localisation comme dans l'approche *Prob-SAILMAP*. Cette quantification se fait habituellement par des techniques de regroupement, tel que k-means[3] et kd-tree[4] pour construire un vocabulaire optimal [158].

3.4 REPRÉSENTATION PAR CARTE TOPOLOGIQUE

Une carte topologique représente l'environnement comme un ensemble de lieux (représentés sous forme de nœuds d'un graphe) qui peuvent être reliés par des arcs. Ces derniers représentent la connectivité qu'il y a entre les lieux, et donc la possibilité de naviguer physiquement et directement entre les nœuds. Une propriété importante et avantageuse des cartes topologiques est leur indépendance à toute information géométrique sur l'environnement. Un exemple classique d'une carte topologique est une carte de métro dans laquelle différentes stations sont représentées sous forme de nœuds, et

3 L'algorithme des k-moyennes (ou K-means en anglais) est un algorithme de partitionnement de données relevant des statistiques et de l'apprentissage automatique (plus précisément de l'apprentissage non supervisé).

4 Un arbre k-d (ou k-d tree, pour k-dimensional tree) est une structure de données de partition de l'espace permettant de stocker des points, et de faire des recherches (recherche par plage, plus proche voisin, ...) plus rapidement qu'en parcourant linéairement le tableau de points.

où les arcs qui connectent des paires de nœuds indiquent la traversabilité. La Figure 3.4 fournit un exemple d'une carte topologique créée dans un environnement intérieur.

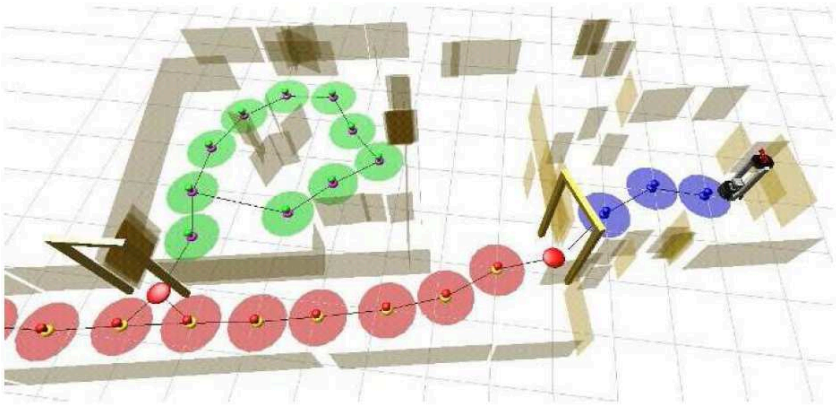


FIGURE 3.4: Un exemple carte topologique construite par un robot [99].

Dans le cadre de ce travail, la carte topologique est construite de façon dense, c'est à dire, que chaque image acquise est représentée par un nœud dans la carte topologique. Chaque nœud de cette carte contient le descripteur V_{F_n}. La carte topologique sera représentée par une mémoire visuelle construite au niveau de l'image.

Dans la première approche *SAILMAP* de détection de fermeture de boucle, ces primitives sont mémorisées de façon brute. Tandis que dans la seconde approche *Prob-SAILMAP*, ces primitives sont quantifiées, afin de réduire la taille de mémoire exigée. En plus de cela et pour un stockage et un appariement de primitives plus efficace, une extension de l'algorithme kd-tree est utilisé. Une estimation bayésienne récursive est finalement appliquée au niveau des images afin d'intégrer l'information de primitives au cours du temps dans un histogramme d'image.

L'architecture du système construisant la mémoire visuelle de l'environnement dans l'approche *Prob-SAILMAP* est illustrée sur la Figure 3.5, cette architecture peut être décomposée en plusieurs étapes. La mémoire visuelle est présentée plus en détails dans la sous-section suivante.

3.4.1 *La mémoire visuelle utilisée dans l'approche ProbSAILMAP*

Les primitives produites par le bloc détecteur/descripteur de saillance sont envoyées à la mémoire visuelle, dont les objectifs sont de représenter l'environnement exploré et de permettre une recherche efficace des primitives similaires précédemment observées. Pour que la localisation soit possible, les primitives doivent être associées à des lieux grâce à la carte topologique, ou au moins à des images. Cela est réalisée par construction de carte topologique. Dans le cas de l'approche *Prob-SAILMAP*, la construction d'une mémoire visuelle fait appelle aux éléments décrit dans les paragraphes suivants.

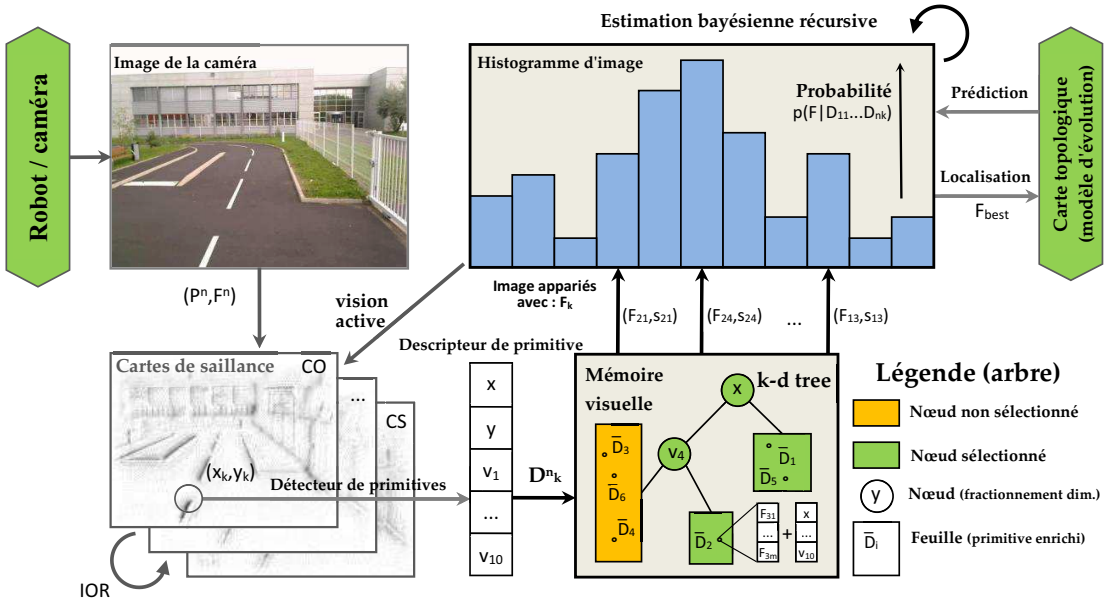


FIGURE 3.5: L'architecture du système de construction de carte. Les composants de bas niveau dont l'extracteur séquentiel de primitive de l'image est affiché sur la gauche, tandis que les composants plus abstraits et cognitifs sont affichés sur la droite. Même si il y a des dépendances entre les composants, la plupart d'entre eux sont intrinsèquement parallèles et peuvent être pipelinés.

3.4.1.1 *Primitives Enrichies*

Les primitives présentées précédemment $\mathbf{D}_k^n$ sont étendues pour définir $\overline{\mathbf{D}}_i$ comme suit :

$$\overline{\mathbf{D}}_i = \left(\mathbf{D}_i, \{F_{ij}\}_{j \in \|1, \phi_i\|}\right) \tag{3.6}$$

où $\{F_{ij}\}$ est l'ensemble des ϕ_i images qui ont été associées à la primitive décrite par $\mathbf{D}_i$. Si une primitive observée est ajoutée à la mémoire visuelle, cela signifie qu'elle n'a pas été appariée avec une primitive préexistante $\overline{\mathbf{D}}_i$. Une nouvelle primitive est ainsi créée, que l'on appellera par la suite *primitive enrichie* : $\mathbf{D}_i = \mathbf{D}_k^n$ et $\{F_{ij}\} = F^n$. Une seule primitive peut être trouvée dans plusieurs images différentes, plus particulièrement lorsque le robot passe par des endroits précédemment visités. Cette redondance est accentuée encore plus par la quantification grossière appliquée lors de la recherche des primitives.

3.4.1.2 *Quantification*

Nous allons considérer que toute primitive extraite $\mathbf{D}_k^n$ qui se trouve dans la zone définie par $[\mathbf{D}_i - \mathbf{R}, \mathbf{D}_i + \mathbf{R}]$ pourrait être associé à $\mathbf{D}_i$, avec $\mathbf{R} = (0.3, 0.3, 0.5, ..., 0.5)$. Si on considère que tous les composants de descripteurs sont normalisés dans $[0, 1]$, ce qui n'est pas toujours garanti concernant l'information saillante en raison de la non-normalisation globale des cartes d'évidence. Cette quantification, choisie pour tester la robustesse de la méthode, peut sembler générer une très faible contrainte de correspondance puisque 0.5 représente la moitié de l'espace.

En raison du fléau de la dimension (Curse of dimensionality)[18], chaque primitive enrichie couvrira moins de 10^{-4} de l'ensemble de l'espace des descripteurs. En d'autres termes, le nombre maximal de primitives enrichies contenu dans l'espace des primitives pour $\mathbf{R} = (0.3, 0.3, 0.5, ..., 0.5)$ est supérieur à 10^6. En pratique et dans le cas de navigation à grande échelle et à long terme, ce nombre est néanmoins relativement faible si chaque lieu n'était représenté que par une seule primitive. Mais puisque des images individuelles sont codées par des ensembles de primitives (seulement jusqu'à 30 dans ce chapitre), nous pouvons représenter idéalement plus de 10^{181} lieux différents avec cette représentation.

Étant donné que de nombreuses primitives avec des coordonnées quantitativement différentes dans l'espace des descripteurs peuvent être associées à la même primitive enrichie, on pourrait envisager la pondération de leurs coordonnées afin de converger vers leur centre de masse. En raison de la complexité supplémentaire introduite par cette adaptation en ligne ainsi que la structure de données présentée ci-dessous (où un partitionnement strict de l'espace est souhaitable pour garder des algorithmes simples et des performances élevées), $\mathbf{D}_i$ prend directement le premier descripteur de la première image qui lui est associée (celui qui a déclenché sa création). Par la suite, seul le numéro de l'image n associé à la primitive observée est ajouté à la liste d'images $\{F_i\}$ de la primitive enrichie $\mathbf{D}_i$ la plus proche trouvée.

3.4.1.3 *La structure kd-tree*

En suivant les descriptions précédentes, afin de rechercher et insérer de manière efficace un nouveau élément dans l'ensemble des primitives enrichies (pouvant avoir jusqu'à 10^6 éléments), la comparaison efficace de vecteurs dans l'espace $\mathbb{R}^{12}$ exige des structures de données dédiées. Une classe spécifique d'arbres de partitionnement binaire de l'espace nommés kd-tree a été choisie car elle permet une recherche d'élément efficace [44]. Pour chaque nœud de l'arbre, l'espace couvert par les nœuds enfants est divisé en deux suivant une seule dimension. Il faut noter que l'architecture kd-tree est ici utilisée pour stocker des primitives, et non pas pour regrouper ou quantifier les primitives pour aboutir à un vocabulaire, comme cela se fait classiquement dans ce domaine [158]. Une fois les primitives ajoutées, peu ou pas de mouvement se produiront entre les nœuds (uniquement pour équilibrer l'arbre si nécessaire).

La complexité du cas le plus défavorable pour la recherche d'un élément dans un arbre standards est connue pour être $\mathcal{O}\left(d.N^{1-\frac{1}{d}}\right)$, Où d est la dimension de l'espace à partitionner [106]. Avec $d = 12$, nous arrivons à une complexité approximative de $\mathcal{O}(d.N)$ qu'un simple tableau pourrait surpasser. Plusieurs extensions ont été développées au fil des ans pour ne pas rencontrer cette limitation. Plusieurs d'entre elles sont implémentées ici en prenant en compte le compromis performance/espace mémoire. L'utilisation des primitives saillantes n'est pas neutre, car la carte de saillance calculée garantit les descripteurs de régions obtenu seront bien répartis dans l'espace. En effet, les étapes de Centre-Contour et de normalisation de cet algorithme sont conçus pour maximiser la visibilité sur un canal à la fois.

3.5 RECHERCHE D'IMAGE PAR SIMILARITÉ

Les architectures et modèles conçus pour cartographier un environnement mémorisent habituellement des images clés et les primitives associées à ces images [38]. Celles-ci sont exploitées dans le processus d'appariement nécessaire à la localisation et/ou la navigation. Pour accroître les performances de ces architectures en résistant aux différents changement qui peuvent se produire, une solution consiste à augmenter le nombre d'images clés, ou le nombre de primitives par images clés. Ce qui conduit inévitablement à un compromis entre la mémoire requise et les performances de calcul. Dans notre cas, uniquement les primitives visuelles sont mémorisées.

3.5.1 *Dans l'approche SAILMAP*

La similarité entre deux images dans l'approche *SAILMAP* peut être estimée en utilisant les descripteurs de RS et les deux contraintes géométriques présentées dans la suite. Soient deux images F_1 et F_2, ayant respectivement m_1 et m_2 RS chacune. Les RS qui sont appariées sont celles qui minimisent la distance euclidienne entre les vecteurs représentant les descripteurs de région ($d_{ij} = \|V_{SR_i} - V_{SR_j}\|_2$). Le coût pour tester tous les couples de RS entre les images ($m_1 \times m_2$) peut être réduit en considérant uniquement les régions qui valident les deux contraintes suivantes :

1. Deux régions appariées ne devraient pas être loin l'une de l'autre, ce qui signifie que pour toutes RS_1 et RS_2 respectivement dans F_1 et F_2, la distance d_{12} est calculée si $\|(x_1, y_1) - (x_2, y_2)\|_2 < \Delta$ (voir Figure 3.6 où le nombre de distances d à calculer est réduit à seulement 2).

2. Les positions relatives des différentes régions dans la première image doivent être approximativement les mêmes dans la seconde image (voir Figure 3.7).

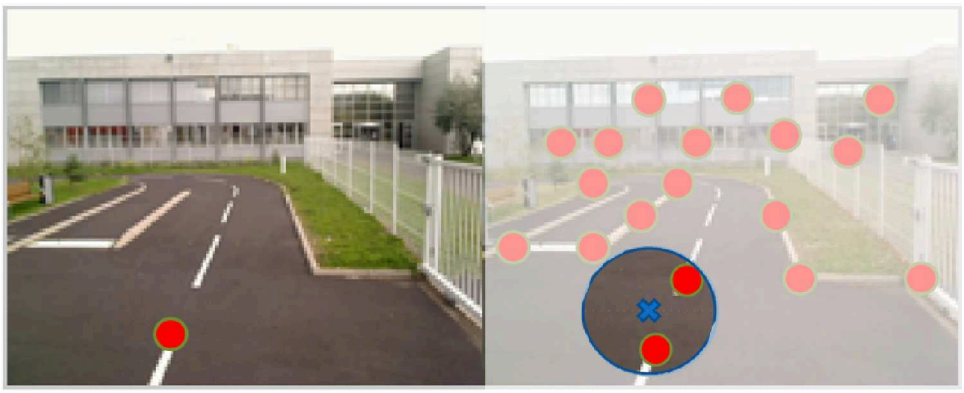


FIGURE 3.6: Primitives dans la zone de recherche, qui satisfont la contrainte de proximité (images de la séquence PAVIN). Ce qui réduit le nombre de distances d à calculer à 2 dans cet exemple.

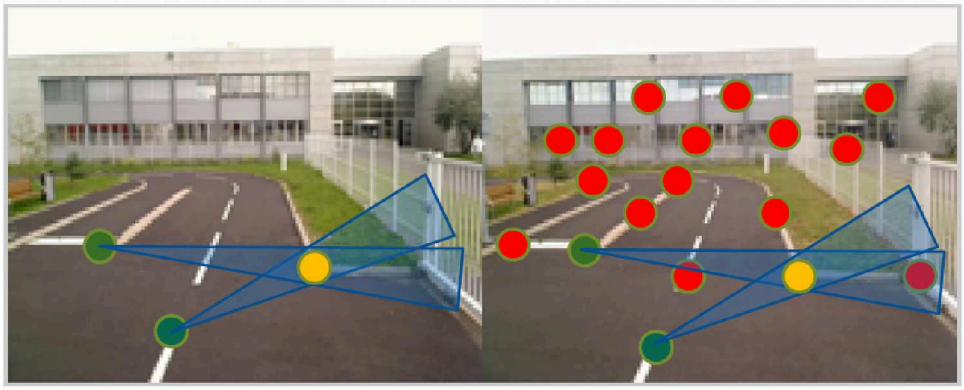


FIGURE 3.7: Primitives pour lesquelles la contrainte sur les positions relatives des RS sont respectées (images de la séquence PAVIN).

Ces deux contraintes permettent d'associer chaque région de F_1 avec un nombre réduit de régions de F_2 ($\ll m_2$). La similarité entre les deux images peut alors être mesurée comme suit :

$$\frac{1}{Sim(V_{F_1}, V_{F_2})} = \frac{1}{|MR|} \times \sum_{(i,j)\in MR} d_{ij} \tag{3.7}$$

$$MR = \{(i,j)/\|(x_i, y_i) - (x_j, y_j)\|_2 < \Delta,\ i = 1 : m_1,\ j = 1 : m_2\} \tag{3.8}$$

où MR est l'ensemble des couples de régions appariées, de sorte que SR_i et SR_j sont respectivement extraites de F_1 et F_2. La similitude est plus élevée lorsque les distances sont plus faible et il y a plus de régions appariées (voir Équation 3.7).

Enfin, $Sim(F_1, F_2)$ est comparée à un seuil prédéfini Th_{LCD}, choisi en adéquation avec la tâche et la précision requise [5]. Ainsi, si $Sim(F_1, F_2)$ est supérieur à Th_{LCD}, les deux images représentent le même lieu, sinon la deuxième image est considérée comme un nouveau lieu.

3.5.2 *Dans l'approche Prob-SAILMAP*

La seconde méthode d'appariement d'images se basant sur l'estimation récursive bayésienne est utilisée dans l'approche *Prob-SAILMAP*. Dans cette approche, les différentes mesures de similarité entre primitives observées et mémorisées sont utilisées pour mettre à jour un histogramme d'images. Celui-ci sera ensuite utilisé pour réaliser la localisation. Cet histogramme d'images est assez similaire à une distribution de probabilité sur les images et il est mis à jour à l'aide d'un filtre d'estimation bayésienne récursif.

Pour toute primitive d'entrée $\mathbf{D}_k^n$, la mémoire visuelle retourne l'ensemble des primitives lui correspondant $\widetilde{\mathbf{D}}_k^n = \{\overline{\mathbf{D}}_i\}_{i\in\|1,\delta_k^n\|}$.Cela nous permet de construire et mettre à jour un histogramme d'images sur la base des votes de chaque primitive enrichie pour ses images associées $\{F_{ij}\}_{j\in\|1,\phi_i\|}$.

5 Pour nos expériences le seuil a été empiriquement choisi pour obtenir les résultats présentés dans la section 3.7 $Th_{LCD} = 20$

Les votes sont calculés comme la similitude entre les descripteurs des primitives observées et les descripteurs de primitives mémorisées :

$$Sim_p\left(\overline{\mathbf{D}}_i, \mathbf{D}_k^n\right) = exp\left(-\frac{\|\mathbf{D}_i - \mathbf{D}_k^n\|^2}{\tilde{o}^2\|\mathbf{R}\|^2}\right) \tag{3.9}$$

Où $\mathbf{R}$ (plage de recherche voir section 3.4.1.2) est utilisée comme un facteur de normalisation. La constante $\tilde{o}$ garantit que le score de similitude sera négligeable pour les primitives en dehors de la zone de recherche, en contribuant à l'écart type de la fonction de similarité gaussienne. La méthode d'insertion et les contraintes de recherche dans l'arbre assurent que les primitives éloignées de celles observées ne participent pas aux votes pour les images dans lesquelles les primitives n'ont jamais réellement été appariées. Un seul vote par image est compté pour une seule primitive observée. Les votes sont normalisés pour tenir compte du pouvoir discriminant des primitives. Les votes associés à chaque primitive enrichie forment donc une distribution de probabilité conditionnelle sur l'espace bâti, noté $p(\mathbf{F}|\mathbf{D}_i)$. L'ensemble $\{\mathbf{D}_i\}$ lui-même peut être interprété comme une approximation éparse de la distribution de probabilité des primitives observées $\mathbf{D}_k^n$ sur la base des similitudes, ce qui conduit à une formulation bayésienne du problème.

3.5.2.1 *Estimation bayésienne récursive*

Soit $\widetilde{\mathbf{D}}^n = \widetilde{\mathbf{D}}_{1:\kappa_n}^n$ l'ensemble de toutes les primitives appariées pour toutes les primitives extraites de l'image F^n. Nous souhaitons donc estimer $p\left(\mathbf{F}^n|\widetilde{\mathbf{D}}^{1:n}\right)$, la probabilité que l'on est effectivement en train d'observer une image précise, ou une autre image qui correspond à un endroit lui ressemblant, tout en intégrant les informations de toutes les primitives observées jusqu'à présent. Cette probabilité conditionnelle sur l'historique des images montre que la navigation peut également être effectuée, car elle introduit une dépendance entre $\mathbf{F}^n$ et $\mathbf{F}^{n-1}, \forall n$.

Étant donné que les primitives sont traitées une à la fois, tout en donnant une estimation continue de la position actuelle du robot, les techniques d'estimations par lot ne sont pas applicables [166]. De même, nous souhaitons détecter une fermeture de boucle au niveau de l'image, et nous n'avons donc pas d'a priori fort sur le type de distributions de probabilités à utiliser. Nous voulons en effet qu'elle puisse être multimodale, excluant ainsi les techniques simples comme le filtre de Kalman. Cependant, nous pouvons toujours calculer $p\left(\mathbf{F}^n|\widetilde{\mathbf{D}}^{1:n}\right)$ en utilisant une estimation bayésienne récursive :

$$p\left(\mathbf{F}^n|\widetilde{\mathbf{D}}^{1:n}\right) \propto p\left(\widetilde{\mathbf{D}}^n|\mathbf{F}^n\right) p\left(\mathbf{F}^n|\widetilde{\mathbf{D}}^{1:n-1}\right) \tag{3.10}$$

Comme les $\mathbf{D}_i$ sont tous indépendants, le premier terme peut être réécrit comme suit :

$$p\left(\widetilde{\mathbf{D}}^n|\mathbf{F}^n\right) = \prod_{k \in \|1,\kappa_n\|} p\left(\widetilde{\mathbf{D}}_k^n|\mathbf{F}^n\right) \tag{3.11}$$

En appliquant la loi de Bayes sur cette dernière équation, la probabilité conditionnelle spécifique à chaque primitive sera définie comme suit :

$$p\left(\widetilde{\mathbf{D}}_k^n|\mathbf{F}^n\right) \propto p\left(\mathbf{F}^n|\widetilde{\mathbf{D}}_k^n\right) p\left(\widetilde{\mathbf{D}}_k^n\right) \tag{3.12}$$

où l'on retrouve les votes décrits dans le paragraphe précédent, ainsi que la distribution a priori basée sur la similitude entre les primitives mémorisées et les primitives observées. Le second terme de l'équation (3.10) peut également être calculé en utilisant l'équation de Chapman-Kolmogorov :

$$p\left(\mathbf{F}_n|\widetilde{\mathbf{D}}^{1:n-1}\right) = \int p\left(\mathbf{F}^n|\mathbf{F}^{n-1}\right) p\left(\mathbf{F}^{n-1}|\widetilde{\mathbf{D}}^{1:n-1}\right) d\mathbf{F}^{n-1} \tag{3.13}$$

Dans laquelle on trouve un modèle de l'évolution du système, qui peut être exprimé en utilisant une matrice stochastique. La connaissance de la vitesse du véhicule par rapport à la fréquence d'acquisition d'images, permet de prédire la séquence d'images attendue et cela juste en décalant l'histogramme d'image actuelle. Enfin, $p\left(\mathbf{F}^{n-1}|\widetilde{\mathbf{D}}^{1:n-1}\right)$ est le terme récursif, estimé pour l'image F^{n-1} qui a déjà été traitée à l'itération précédente. Toutes les probabilités qui ne sont pas affichées dans les équations peuvent être évaluées en utilisant le contenu de l'arbre kd-tree. Pour accélérer certains calculs l'approche d'index inversé peut être utilisée, par exemple pour estimer le nombre de primitives ayant étés stockées pour une image donnée.

Pour l'initialisation en navigation ou pour la localisation, on peut simplement choisir une distribution uniforme pour la probabilité a priori sur les images. Étant donné qu'une image ne peut pas être directement représentée dans l'histogramme avant d'être observée, la probabilité F^n doit être bien différenciée de $\mathbf{F}^n$, qui reflète toutes les fermetures de boucle possibles à partir des images déjà mémorisées $\{F^i\}_{i\in\|1,n-1\|}$. Pour la localisation, le maximum de vraisemblance peut être adopté pour prendre une décision, mais il ne fonctionne bien que lorsque le filtre bayésien intègre le modèle d'évolution entre des images successives (par lissage et nettoyage de l'histogramme). Nous avons donc opté pour une somme pondérée sur l'espace des images, avec une fenêtre *ws* de 4 images autour du maximum. Le codage par population utilisé dans l'histogramme permet donc à la valeur décimale résultante de l'estimation du numéro de l'image d'aller au-delà de la résolution de l'arbre (gardé faible pour limiter les besoins en mémoire), et de l'échantillonnage temporel fixé pour l'acquisition des images.

3.6 FERMETURE DE BOUCLE VS LOCALISATION

Lorsque le robot explore une nouvelle région ou lorsqu'une séquence d'images est fournie au système, le descripteur global de chaque nouvelle image traitée est comparé aux descripteurs déjà présents dans les nœuds pour la détection d'une éventuelle fermeture de boucle.

Pour cela, il faut bien distinguer le problème de localisation et celui de détection de fermeture de boucle. Dans le premier, l'image de test doit être mise en correspondance avec une des images déjà présente dans la carte, tandis que dans le second problème, le système doit prendre une décision concernant les images traitées afin de déterminer si elles proviennent de lieux déjà visités ou bien s'il faut les considérer comme de nouveaux lieux.

La détection de fermeture de boucle est faite en appariant les RS de l'image courante avec les RS qui sont dans les nœuds de la carte topologique. Cela signifie que la similarité est calculée entre le descripteur d'image en cours et l'ensemble des descripteurs attachés aux nœuds.

Dans l'approche *SAILMAP*, après l'acquisition d'une image et le calcul de toutes les similitudes, le maximum est trouvé et comparé à $\mathrm{Th}_{\mathrm{LCD}}$. Si la valeur maximum est supérieure à $\mathrm{Th}_{\mathrm{LCD}}$, le système mémorise simplement le nœud correspondant (défini comme actif). Si par contre la valeur est inférieure à $\mathrm{Th}_{\mathrm{LCD}}$, un nouveau nœud est rajouté à la carte avec le descripteur associé à l'image courante, un nouvel arc est créé pour connecter le nœud préalablement activé au nouveau nœud, qui est finalement défini comme actif (voir Figure 3.4). Pour la première image, le descripteur est automatiquement ajouté à la carte en tant que premier nœud.

Dans la seconde approche *Prob-SAILMAP*, après l'acquisition d'une image, cette dernière est traitée par l'algorithme de saillance qui détecte les primitives dans l'ordre jusqu'à ce qu'une nouvelle image arrive. Pour chaque primitive détectée, un descripteur de l'attention est calculé et transféré à la mémoire visuelle. La mémoire visuelle cherche des primitives similaires qui auraient déjà été capturées, sinon si la primitive est nouvelle elle sera ajoutée ultérieurement à la mémoire visuelle. L'histogramme d'image est ensuite mis à jour en fonction de la similitude entre les primitives observées et appariées. Si l'histogramme répond à des critères spécifiques, il peut déclencher un événement de detection de fermeture de boucle et mettre à jour la carte topologique associée représentant l'environnement, ou prédire les primitives suivantes à détecter (*perception active*).

3.7 RÉSULTATS EXPÉRIMENTAUX

Les résultats présentés dans cette section englobent des tests de localisation et détection de fermeture de boucle lors de la construction de carte topologique, en utilisant des primitives de type saillance.

Les premières expérimentations concernent uniquement l'utilisation de la première approche *SAILMAP*. Les résultats obtenus avec cette approche sont comparés aux résultats obtenus avec l'algorithme de Cummins and Newman (2008) [41] connu sous le nom de FABMAP, et qui fait actuellement référence dans le domaine. Dans cet algorithme les primitives utilisées sont les SIFT, combinées avec la technique de représentation par sac-de-mots (BoW) pour quantifier les différentes primitives. La seconde partie de cette section donne les différents résultats obtenus lorsque l'approche probabiliste (*Prob-SAILMAP*) est utilisée. L'exactitude des différents résultats est mesurée à l'aide des deux paramètres *Précision* et *Rappel*, qui sont définis comme suit :

$$\mathrm{Precision} = \frac{\mathrm{FBD}(\mathrm{Positifs})}{\mathrm{FBD}(\mathrm{Positifs} + \mathrm{Négatifs})} \tag{3.14}$$

$$\mathrm{Rappel} = \frac{\mathrm{FBD}(\mathrm{Positifs})}{\mathrm{FB}(\mathrm{Vérité\ Terrain})} \tag{3.15}$$

FBD : Fermeture de Boucle Détecté.

Les expérimentations ont été effectuées sur différents ensembles de données, deux de l'université d'Oxford (Royaume-Uni) et deux de l'Institut Pascal (France). La Figure 3.3 montre une vue de dessus des quatre sites (New College - City Center - PAVIN - CEZEAUX).

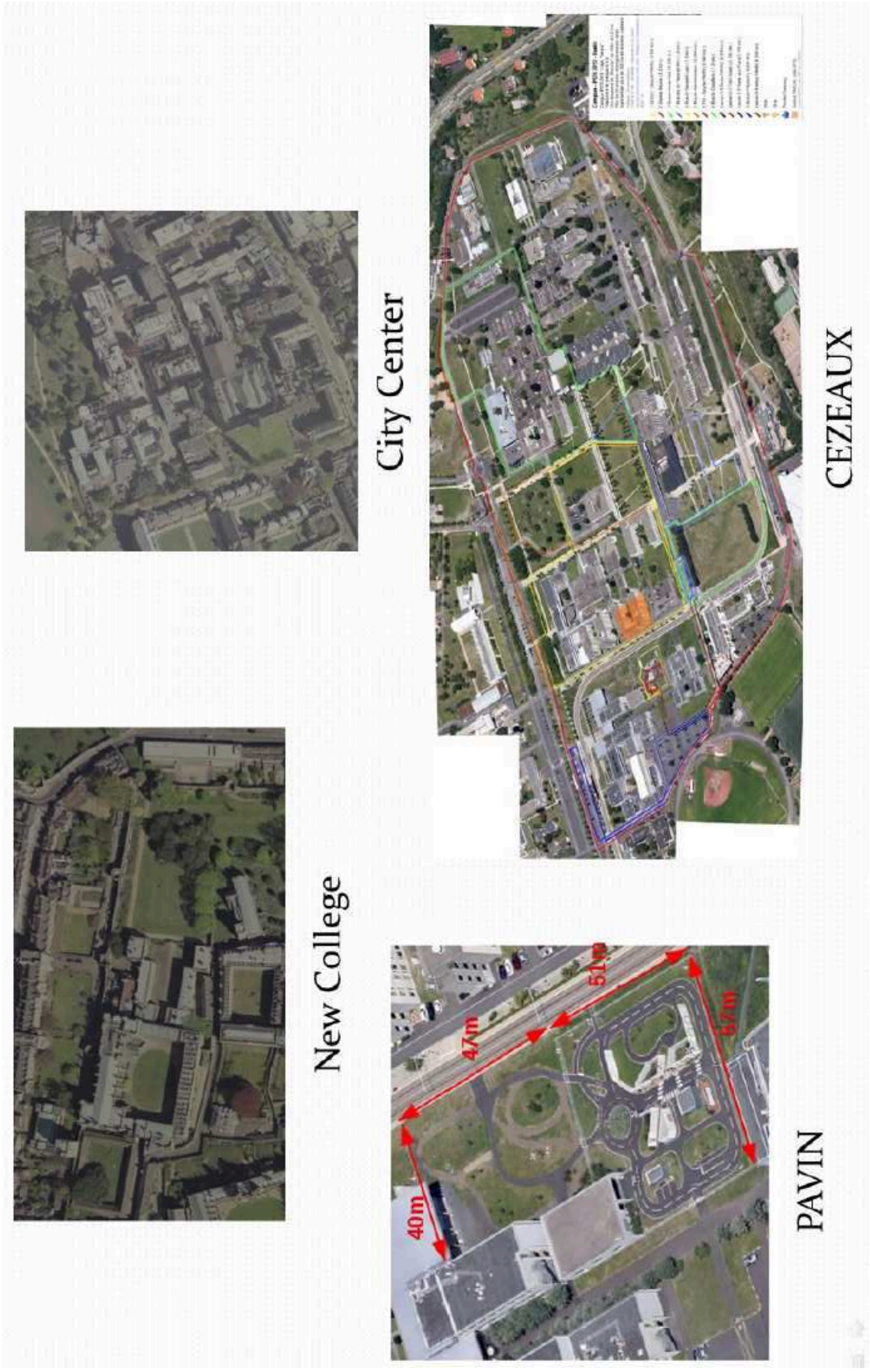


FIGURE 3.8: Les quatre sites où les ensembles de données ont été acquis (images de Google-Map).

3.7.1 *Les ensembles de données*

Les ensembles de données de l'université d'Oxford ont été recueillis à l'aide d'un robot mobile dans un environnement extérieur. Le robot recueille des images à gauche et à droite le long de sa trajectoire approximativement tous les 1.5m.

La première série de données New College (NC) (1.9km de long) a été choisie pour tester la robustesse du système pour l'aliasing perceptuel. Elle dispose de plusieurs grandes zones de forte répétition visuelle, y compris un cloître médiéval avec la répétition de nombreuses arcades identiques et un jardin avec un long tronçon de mur et des buissons uniformes. La deuxième série de données étiquetées City Center (CC) (2km de long) a été choisie pour tester la capacité de mise en correspondance des images en présence d'objets dynamiques. Elle a été recueillie le long des routes publiques à proximité du centre ville, et propose de nombreux objets dynamiques tels que les voitures et les piétons. En outre, les deux séries ont été recueillies sur une journée venteuse avec un soleil radieux, ce qui rend le feuillage et les ombres caractéristiques abondants et instables.

Dans les séries de données de l'Institut Pascal deux séquences d'images sont utilisées[6]. Les séries de données sont acquises en utilisant la plate forme *VIPALAB*[7]. Les images en couleurs sont acquise par la webcam Logitech QuickCam Pro-9000, embarquée sur le toit du véhicule et regardant devant. Il existe également un système GPS qui fournit une mesure de localisation absolue, qui ne sert que comme une vérité terrain pour la validation des différentes fermetures de boucle détectées et pour vérifier l'exactitude du processus de localisation.

Les deux séquences PAVIN (PV) et CEZEAUX (CZ) portent les noms des sites d'acquisition. *PAVIN* est un site artificiel qui simule des environnements urbains et ruraux avec des routes goudronnées, des marquages routier, des feux, des plaques de signalisation, des ronds-points et des carrefours. *CEZEAUX* est la région environnante l'Institut Pascal et l'Université Blaise Pascal, c'est un environnement de style semi urbain avec des routes, des bâtiments, des pelouses et de la végétation sur les deux côtés de la route. En raison de la grande taille de l'environnement, il a été impossible de naviguer à travers chaque voie, mais les séquences acquises couvrent tous les lieux les plus importants du site.

Dans les deux sites (*PAVIN* et *CEZEAUX*), six séquences ont été acquises. Elles sont illustrées dans la Figure 3.9 générée en utilisant les données GPS des différentes séquences correspondantes. Chaque séquence est présentée séparément dans la Figure 3.10 et divers détails peuvent être trouvées dans le Tableau 3.2.

PV-Jonco et *CZ-Sealiz* sont les séquences principales les plus longues, elles contiennent plusieurs retraversées (fermeture de boucle). Cependant, *PV-Jonco* est une séquence exploration c'est à dire que tous les chemin de l'environnement sont traversés. *PV-Elo* et *CZ-Heko* sont des séquences courtes de *PAVIN* et *CEZEAUX*. Ces dernières peuvent être utilisées pour l'apprentissage du vocabulaire visuel. La particularité de la séquence *PV-Hema* est qu'elle est acquise à une vitesse variable. Ce qui veut dire que les retraversées se produisent à une vitesse différente de celle des traversées précédentes. Cette

6 Les ensembles de données sont hébergés sur http://ipds.univ-bpclermont.fr/.
7 VIPALAB est un véhicule conçu pour servir de prototype pour la recherche et le développement dans le domaine des véhicules de transports urbains autonomes.

FIGURE 3.9: Six séquences de PAVIN et CEZEAUX tracées avec les données GPS.

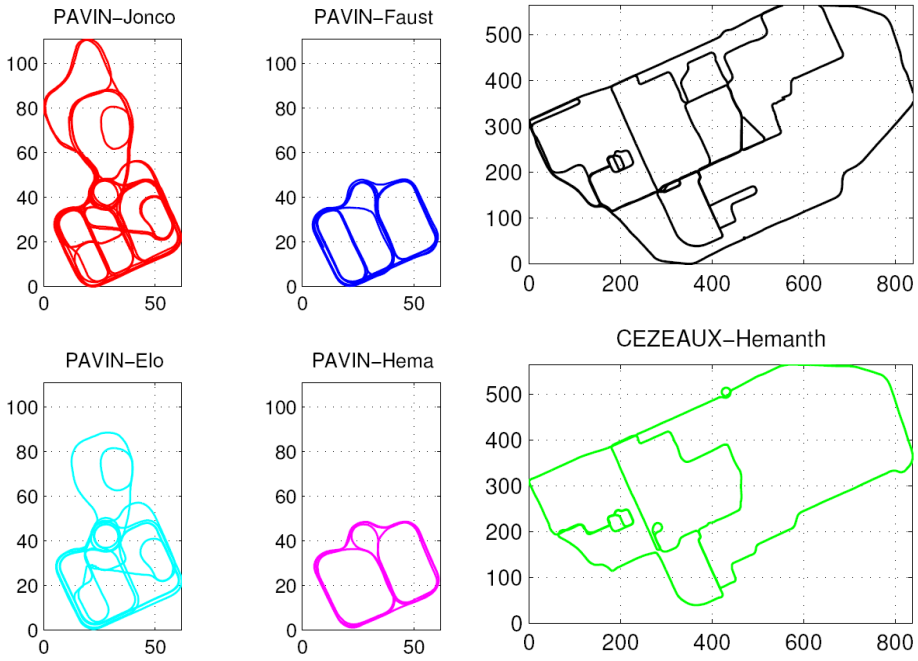
séquence est particulièrement utile pour tester les limites des approches de détection de fermeture de boucle qui supposent une vitesse constante et pour évaluer les approches qui utilisent un modèle de mouvement pour capturer les retraversées avec une vitesse différente. *PV-Faust* contient de longues boucles allant de 85m à 160m. Ces longues boucles peuvent trouver une application dans les algorithmes de SLAM pour évaluer la robustesse de la dérive de l'odométrie.

Plus de détails au sujet de ces séquences peuvent être trouvées sur le site web : `http://ipds.univ-bpclermont.fr/`.

SÉQUENCE	LONGUEUR (KM)	DURÉE (MIN)	TAILLE DES DONNÉES (GO)	VITESSE MOYENNE (M/S)	NOMBRE D'IMAGES	
					WEBCAM	FRONT-LEFT-RIGHT-CAM
PAVIN-Jonco	2.3	18	90.0	2.2	16'330	8'097
PAVIN-Elo	1.2	12	64.6	0-6	10'988	5'453
PAVIN-Hema	0.6	5	26.8	2.2	4'566	2'330
PAVIN-Faust	1.3	10	54.3	2.2	9'247	4'623
CEZEAUX-Sealiz	7.8	52	269.5	2.5	45'923	23'027
CEZEAUX-Heko	4.2	28	150.5	2.5	25'680	12'853

TABLE 3.2: Les séquences de données de l'IPDS.

La Figure 3.12 donne un aperçu du résultat obtenu pour les deux approches SAILMAP et FABMAP lorsque les courbes Précision/Rappel sont étudiées sur deux ensembles de données. Une précision de 100% peut être obtenue avec l'approche SAILMAP sur les deux ensembles, contrairement à FABMAP, où une précision de 100% est obtenue uniquement sur le second ensemble.

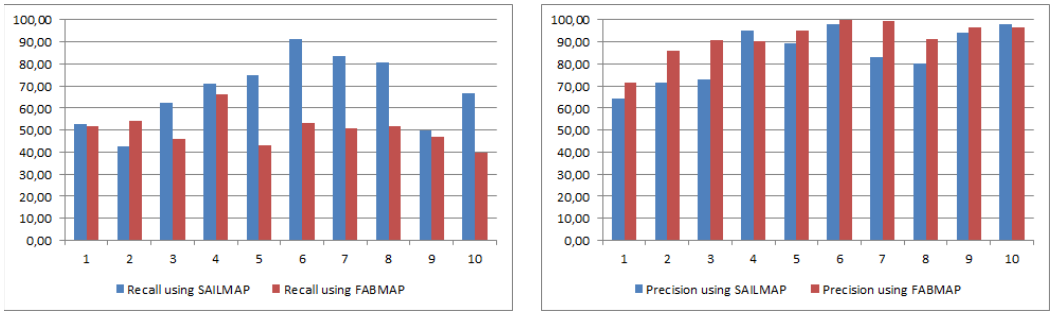


(a) Quatre séquences de PAVIN. (b) Deux séquences de CEZEAUX.

FIGURE 3.10: Les six séquences illustrées séparément.

3.7.2 *L'approche SAILMAP*

Le Tableau 3.3 donne les résultats obtenus pour la détection de fermeture de boucle lors de l'utilisation de l'approche FABMAP et *SAILMAP* respectivement sur les différents ensembles de données. Les résultats de *Précision* et de *Rappel* sont également présentés sur la Figure 3.11.



(a) Rappel. (b) Précision.

FIGURE 3.11: SAILMAP vs FABMAP sur les séquences : 1-CC-Left, 2-CC-Right, 3-NC-Left, 4-NC-Right, 5-PV-Heman, 6-PV-Faust, 7-PV-Elo, 8-PV-Jonco, 9-CZ-Heko, 10-CZ-Sealiz.

Sur la Figure 3.11a il est clair que l'approche *SAILMAP* donne des scores pour le *Rappel* supérieur à ceux obtenus avec l'approche FABMAP, à la seule exception de la séquence CC-*Right*. La différence moyenne du score de précision sur les différentes séquences est de 16%. Dans la séquence PV-*Faust* le *Rappel* obtenu en utilisant *SAILMAP* est égal à 91%, alors qu'avec FABMAP il est égal à 53% seulement, ce qui montre que *SAILMAP* est capable de détecter plus de fermetures de boucle, et même si avec FABMAP

	PRÉCISION (%)			RAPPEL (%)		
SÉQUENCE	FABMAP	SAILMAP	DIFF	FABMAP	SAILMAP	DIFF
CC-Left	**71**	64	-7	52	**53**	+1
CC-Right	**86**	72	-14	**54**	43	-11
NC-Left	**91**	73	-18	46	**62**	+16
NC-Right	90	**95**	+5	66	**71**	+5
PV-Heman	**95**	90	-5	43	**75**	+32
PV-Faust	**100**	98	-2	53	**91**	+38
PV-Elo	**99**	83	-16	51	**84**	+33
PV-Jonco	**91**	80	-11	52	**81**	+29
CZ-Heko	**97**	94	-3	47	**50**	+3
CZ-Sealiz	96	**98**	+2	40	**67**	+27

TABLE 3.3: Les résultats de « *Précision* » et de « *Rappel* » obtenus en utilisant *FABMAP* et *SAILMAP* sur les différents ensembles de données.

une précision de 100% est obtenu, un score de 98% est obtenu en utilisant *SAILMAP*, ce qui est également un très bon résultat. En revanche, comme le montre la Figure 3.11b représentant la *Précision*, les résultats obtenus avec l'approche *SAILMAP* sont proches de ceux obtenus avec l'approche FABMAP, et la différence moyenne sur les différents ensembles de données est de seulement 7%.

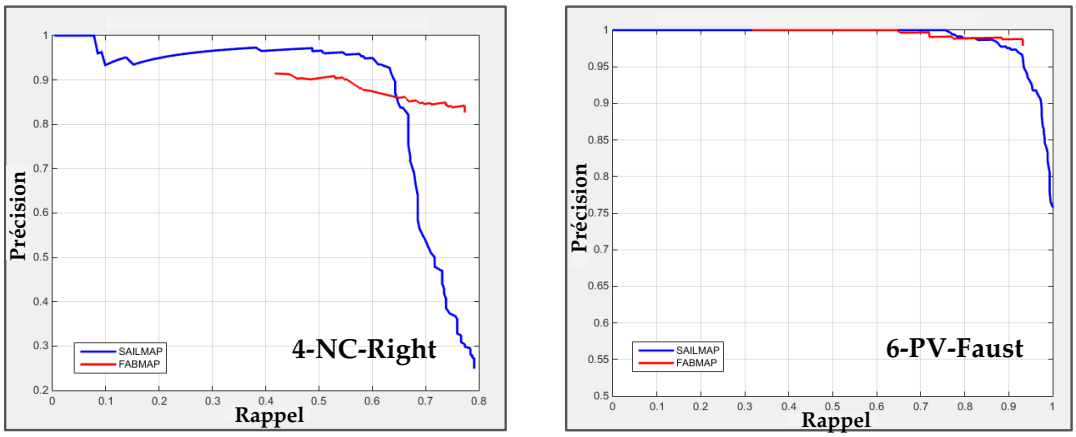


FIGURE 3.12: Courbes Précision=f(Rappel) pour les deux ensembles de données NC-Right et PV-Faust.

3.7.3 *L'approche Prob-SAILMAP*

Les résultats présentés dans cette partie correspondent uniquement à la base de données « PV-Elo » (1284m ; 12min ; 1,75m/s). Une image sur 12 a été utilisée afin de rendre moins dense la base de données et ainsi complexifier davantage la localisation ($10988/12 = 917$ images, avec une carte de saillance de 160×120 pixels). Bien que l'apprentissage et la localisation puissent être effectués simultanément, nous nous concentrons ici sur des scénarios spécifiques où l'arbre (kd-tree) est autorisé à croître pendant

une séquence fixe d'images, puis est figé pour tester ses performances sur une autre série d'images. Les scénarios considérés pour valider l'approche sont les suivants

- **Reconnaissance** : l'apprentissage se fait sur les images 1 à 10998 (par pas de 12 images), et la localisation sur les mêmes images. Cela permet de tester simplement la capacité de la mémoire visuelle à distinguer les images malgré la forte quantification des ensembles de primitives. Les deux valeurs d'erreur visible sur le Tableau 3.4 correspondent à la présence (45cm) ou absence (6cm) d'interpolation pour prendre une décision.

- **Interpolation** : l'apprentissage est également effectuée sur les images $[1:12:10998]$, mais la localisation est testé sur $[K:12:10998]$. Lorsque K augmente dans $[2,6]$, l'erreur augmente de façon exponentielle. L'erreur affichée dans le Tableau 3.4 correspond à $K=6$ (pire cas). Étant donné que la trajectoire passe plusieurs fois par les mêmes endroits mais avec des trajectoires différentes, ce scénario montre que les images sont globalement bien appariées, surtout en mode navigation.

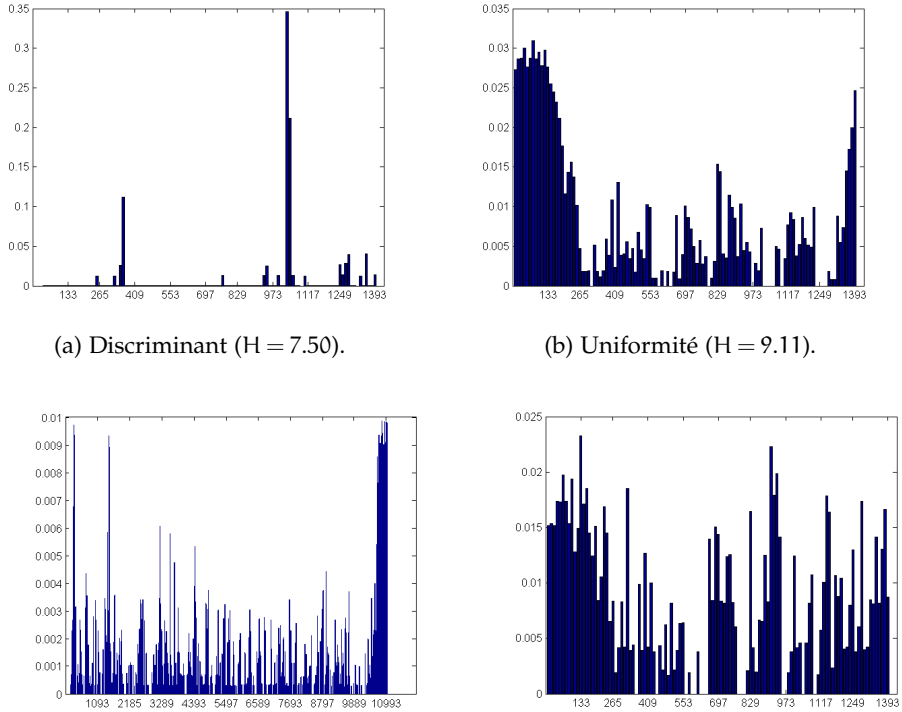
- **Généralisation** : l'apprentissage ne se fait que sur les images $1:12:1440$ (premier tour autour de la plate forme urbaine PAVIN). La localisation est testée sur tous les segments où l'emplacement et l'orientation sont plus ou moins partagées avec l'ensemble des images utilisées en phase d'apprentissage (même si sur différentes voies/côté de la route), par exemple le début du deuxième tour de $1441:12:1640$. Les chiffres entre parenthèses correspondent à une augmentation de lissage ($ws=8$), ce qui reflète la tendance que la décision devrait être plus distribuée si le niveau de bruit augmente pour chaque mesure (voir la Figure 3.13).

Les résultats de la précision de localisation et de détection de fermeture de boucle sont reportés dans le Tableau 3.4. L'erreur est donnée par la distance en mètres entre l'emplacement du robot pour l'image fournie en entrée et l'emplacement interpolé du robot pour les images estimées. Cette erreur est calculée à partir des coordonnées GPS (vérité terrain) associées aux images mémorisées. Ces coordonnées ne sont évidemment pas fournies à l'algorithme pour effectuer la localisation. Bien que l'erreur de localisation pure soit assez élevée en allant vers les zones antérieures tout en prenant une voie légèrement différente, l'impact de la navigation doit être souligné car elle réduit considérablement l'erreur. Cette dernière doit encore être comparée à la résolution spatiale maximale de l'entrée de l'appareil fourni, avec une distance moyenne entre les images successives de 1.4m (voir Figure 3.14).

SCÉNARIO	RECONNAISSANCE 1 :12 :10998		INTERPOLATION 6 :12 :10998		GÉNÉRALISATION 1400 :12 :10998	
Mesure	Erreur (m)	%	Erreur (m)	%	Erreur (m)	%
Localisation	0.45 (0.06)	100	3.50	77	3.19 (3.25)	33 (40)
Navigation	0.16 (0.00)	100	0.39	99	1.65 (1.09)	71 (94)

TABLE 3.4: Erreur et précision pour différents scénarios/configurations.

En termes de mémoire et afin de représenter la trajectoire de 1.2 km, l'arbre crée 841 nœuds ou feuilles, et a une profondeur égale à 14. Les feuilles contiennent au final 2693 primitives enrichies, qui sont associées aux 27510 primitives observées lors de la phase d'apprentissage sur la séquence $[1:12:10998]$ (917 images $\times 30$ primitives/image). La structure atteint un taux de compression de données égale à 10, la grande majorité



(a) Discriminant (H = 7.50).

(b) Uniformité (H = 9.11).

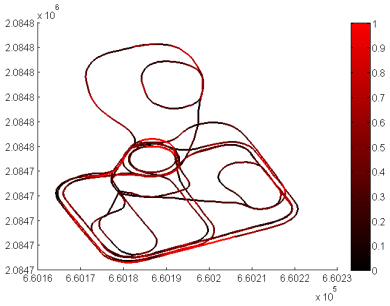
(c) Fermeture de boucle (H = 8.47).

(d) Ambiguïté (H = 9.35).

FIGURE 3.13: Histogrammes d'images typiques. Bon appariement en (a) et (c) correspondant à une entropie faible (donnée entre parenthèses), avec détection multiple de fermeture de boucle dans (c) lors de l'utilisation de l'histogramme complet (10998 images). Il n'y a pas de pics clairs dans (b), parce que le véhicule est statique, et le véhicule est dans un endroit totalement inconnu en (d).

des descripteurs de primitive d'origine (12 valeurs réelles) étant remplacée par l'unique numéro de l'image. Le processus complet de recherche et la mise à jour d'histogramme s'exécute en 17ms/primitive sur un ordinateur portable équipé d'un processeur Quad-Core i7Q740, sous Matlab-2014.

(a) La superposition des images correspondant à l'erreur maximum GPS.



(b) Trajectoire entièrement reconstruite à partir des données du GPS (Erreur du GPS en couleur).

FIGURE 3.14: Illustration graphique de l'erreur pour le scénario d'interpolation.

3.8 CONCLUSION

L'approche *SAILMAP* proposée dans ce chapitre peut être utilisée pour construire des cartes topologiques dense ou éparse, comme elle peut être utilisée pour faire de la localisation avec ou sans a priori, ou bien encore de la détection de fermeture de boucle.

Le descripteur proposé n'est pas affecté par les différents changements qui peuvent se produire dans le fond de la scène contrairement à celui proposé par Frintrop (2006) [56], pour la simple raison que dans notre approche, chaque élément est calculé localement. Il faut noter aussi que le même calcul est fait pour la phase de détection et de description.

Enfin, les éléments de l'architecture proposée sont intrinsèquement et massivement parallélisable. La seconde partie de ce manuscrit présente les architectures de traitement d'images embarquées reconfigurables favorisant les traitements parallèles. Ainsi que l'implémentation matérielle de l'algorithme de détection de régions saillantes sur une de ces architectures.

Troisième partie

ARCHITECTURE

4

ARCHITECTURE DE TRAITEMENT D'IMAGES EMBARQUÉE

4.1 INTRODUCTION

Une tendance récente dans plusieurs tâches de la robotique est de considérer la vision comme l'élément essentiel avec lequel la perception de l'environnement ou l'interaction avec d'autres utilisateurs peut se réaliser. C'est pour cela que le traitement d'images a connu un développement considérable au cours de cette dernière décennie.

Souvent en robotique les différents traitements doivent être réalisés par des systèmes de traitement temps réel. De tels systèmes peuvent être obtenus en optant pour des solutions matérielles, par exemple à base de FPGA. Bien que ces derniers aient une fréquence d'horloge faible, ils restent néanmoins des composants puissants surpassant les autres systèmes lorsque le parallélisme de traitement peut être exploité.

L'objectif de ce chapitre est de présenter le domaine d'application dans lequel nous travaillons. Nous nous intéressons plus particulièrement aux différentes architectures embarquées de traitements numériques des images en temps réel. Pour cette raison, nous allons tout d'abord présenter la chaîne de traitement d'images standard et la classification des algorithmes sous trois niveaux d'abstraction. Ensuite, nous rappellerons l'architecture des composants FPGA utilisés dans la conception des différentes plates-formes reconfigurables permettant l'implémentation matérielle des algorithmes de traitement d'images. Avant de terminer ce chapitre, nous établirons un état de l'art des différentes architectures que l'on peut trouver dans la littérature et qui ont l'avantage d'être particulièrement adaptées au domaine du traitement d'images.

4.2 CHAÎNE DE TRAITEMENT D'IMAGES

Actuellement le domaine d'application du traitement d'images est vaste et varié, comme par exemple la vision industrielle, l'imagerie aérienne et spatiale, l'analyse médicale, la robotique ... Les tâches principales réalisées dans ces différents domaines sont le contrôle, l'inspection et l'acquisition de données. L'objectif du traitement d'images est d'établir une liste de caractéristiques des scènes perçus dans l'image, afin de guider un processus ou prendre une décision. Traditionnellement, on décompose le traitement d'image en trois classes d'abstraction : bas, moyen et haut niveau [191]. La Figure 4.1 présente la chaîne de traitement d'une image.

4.2.1 *Le traitement bas niveau*

L'entrée du traitement d'image bas niveau est faite par un capteur de type Charge Coupled Device (CCD) ou Complementary Metal Oxide Semiconductor (CMOS). Ce capteur

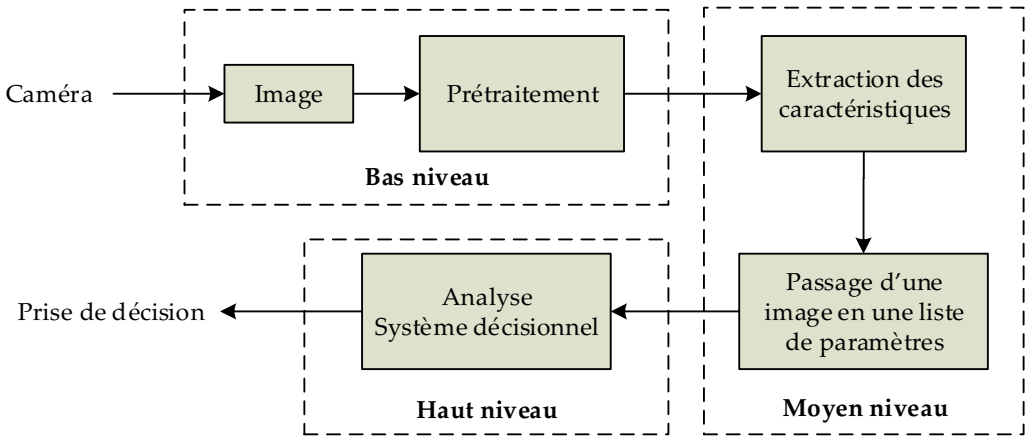


FIGURE 4.1: Traitement numérique d'images en trois niveaux d'abstraction.

permet l'acquisition de l'image et à l'aide d'un dispositif de conversion, on passe du domaine analogique au numérique. Le bloc de traitement suivant consiste à améliorer l'image, en élimant le bruit parasite et en améliorant le contraste de l'image. Les opérations utilisées dans cette étape sont généralement locales, c'est l'étape de *prétraitement* d'image. Le résultat de ce niveau est une image. Exemple : filtrage, corrélation, convolution, ...

4.2.2 *Le traitement moyen niveau*

L'objectif de ce traitement est l'extraction de primitives à partir des images fournies par le traitement bas niveau. Une première étape consiste à extraire de l'image des caractéristiques de type points, contours, régions ou textures. Les algorithmes utilisés pour l'extraction de ce type de caractéristiques sont composés d'une succession d'algorithmes. Par exemple, lors de la détection des régions saillantes, il est nécessaire d'utiliser en premier la pyramide dyadique afin d'obtenir des images à multi-échelle, d'appliquer un algorithme de soustraction *Centre-Contour* et puis d'autres algorithmes séquentiellement (voir le chapitre 2), pour au final trouver la liste contenant les coordonnées des régions saillantes de l'image.

Le passage d'une image à une liste de paramètres est l'étape finale dans ce niveau de traitement d'images. Ce niveau de traitement est utilisé pour réduire la quantité de données à stocké ou à transmettre. Les algorithmes appartenant à ce niveau sont plus complexes que ceux du bas niveau, ce qui nécessite l'utilisation de plates-formes de calculs puissantes. Le résultat de cette étape est une liste de primitives (points, droites, régions, textures, ...)

4.2.3 *Le traitement haut niveau*

A ce niveau, c'est la prise de décision qui est faite, en exploitant les caractéristiques issues du processus de moyen niveau. Cette tâche de prise de décision est réalisée en comparant les caractéristiques extraites dans le traitement moyen niveau avec les caractéristiques déjà présentes dans une base de données.

Par exemple, dans le domaine de la robotique mobile, des systèmes complets utilisant la vision ont été mis en place pour détecter les fermetures de boucles lorsque l'on cartographie un environnement. Il est alors nécessaire d'extraire de la scène actuelle des caractéristiques, ensuite de les comparer avec les caractéristiques d'une base de données. Une décision est par la suite prise sur la présence ou non d'une fermeture de boucle.

Dans le Tableau 4.1, nous présentons une liste (non exhaustive) d'algorithmes que nous classifiant suivant les trois niveaux d'abstraction.

TRAITEMENT D'IMAGES	ALGORITHMES
Bas niveau : Prétraitement.	Opérateurs de seuillage, convolution, estimation de mouvement, calcul de flot optique, histogramme, lissage, filtres passe-bas, filtres médians, filtres morphologiques, filtres de contours, transformée de Fourier, transformée en ondelettes, transformée de Hough, ...
Moyen niveau : Extraction des caractéristiques et passage d'une image en une liste de paramètres.	Points d'intérêt, régions saillante, texture, mouvement, profondeur, ...
Haut niveau : Système décisionnel.	Classification, algorithme d'optimisation, similarité, homogénéité, ...

TABLE 4.1: Quelques exemples d'algorithmes de traitement d'images.

4.3 PLATES-FORMES RECONFIGURABLES POUR LE TRAITEMENT D'IMAGES

4.3.1 *Introduction*

La quantité de données à manipuler en traitement d'images et les opérations nécessaires (souvent répétitifs), même pour un calcul simple font appel à une puissance de calcul importante, notamment lorsque la contrainte temps réel est exigée. Parmi les solutions qu'on trouve dans la littérature pour satisfaire la contrainte du temps réel : il existe des architectures figée à base de Digital Signal Processing (DSP) et Graphics Processing Unit (GPU), et des architectures reconfigurables à base de FPGA.

Dans nos travaux, nous nous sommes uniquement intéressés aux plates-formes reconfigurables à base de FPGA. Les FPGA offrent un degré élevé de flexibilité et de performance pour gérer de nombreuses applications, comme par exemple : la vision stéréo [87], l'algèbre géométrique [54], le flux optique [192], la reconnaissance d'objets [121] ou la vidéo-surveillance [137, 165] ...

Inventés en 1985 par Xilinx [2], les circuits FPGA ont connu une rapide évolution depuis plusieurs années. Le premier circuit FPGA commercial est le XC2000 de Xilinx. Ce composant avait une capacité maximum de 1500 portes logiques, utilisant la technologie aluminium à 2µm. Altera lança en 1992 la famille de FPGA FLEX8000 dont la capacité maximum atteignait 15000 portes logiques.

Actuellement sur le marché, Xilinx propose une nouvelle génération de FPGA nommée *Series 7* utilisant la technologie 28nm High Performance Low power (HPL)(voir [86]). Ce qui permet aux concepteurs de maximiser la productivité et d'améliorer les performances des traitement tout en ayant une faible consommation d'énergie. En Mai 2014, Xilinx a lancé une autre génération, l'*UltraScale* series composée de : Virtex UltraScale et Kintex UltraScale. Ces nouvelles familles de FPGA sont fabriqués par Taiwan Semiconductor Manufacturing Company (TSMC), en utilisant la technologie 20nm.

Quat aux composants FPGA proposés par Altera, on trouve trois familles : STRATIX (haut gamme), ARRIA (milieu de gamme) et CYCLONE (entrée de gamme). Les derniers FPGA apparus entre 2010 et 2012 de chaque famille sont fabriqués en utilisant la technologie 28nm. En 2013, Altera lance les nouvelles architectures System on Chip (SoC)-FPGA, qui offrent de nombreuses fonctionnalités révolutionnaires, y compris une architecture haute performance avec la technologie 20nm. Dans ce qui suit, nous présentons l'architecture des FPGA et les nouvelles architectures SoC-FPGA.

4.3.2 *Architecture des FPGA*

Un FPGA est un dispositif dont l'architecture interne est configurée en fonction de l'application à exécuter. Contrairement aux DSP et GPU, qui ont la particularité d'être des composants dont l'architecture interne est figée.

La Figure 4.2 représente l'architecture générale des FPGA, elle est essentiellement constituée de trois éléments :

- **Les Configurable Logic Block (CLB)** : Constituent le cœur du FPGA. Ce sont des cellules constituées d'éléments logiques programmables où l'on trouve des bascules, des Look-Up Table (LUT), des multiplexeurs et des portes logiques disposées sous forme matricielle, comme le montre la Figure 4.2. Chaque cellule est identique aux autres et peut-être reliée à ses voisins par le biais de bus d'interconnexion. Ces cellules peuvent être utilisées pour créer des fonctions logiques complexes, mais également comme éléments de stockage de variables. Grâce à l'évolution de la technologie, l'architecture de ces cellules a évolué en complexité. Par ailleurs, les FPGA intègrent de plus en plus de cellules sur une même surface. Par exemple, jusqu'à 400'000 CLB appelés high performance Adaptation Logic Modules (ALM) dans les Stratix-V de Altera [1] et 1'956'000 CLB pour les Virtex-7 de Xilinx [2].

- **Les Input Output Block (IOB)** : Ces cellules d'entrées-sorties permettent d'interfacer le FPGA avec l'environnement extérieur. Chaque bloc IOB contrôle une broche du composant et il peut être défini en entrée, en sortie, en signal bidirectionnel ou être inutilisé (haute impédance).

- **Les ressources d'interconnexion des cellules** : Pour pouvoir réaliser des fonctions complexes à partir des fonctions de base que représentent les CLB, il est nécessaire

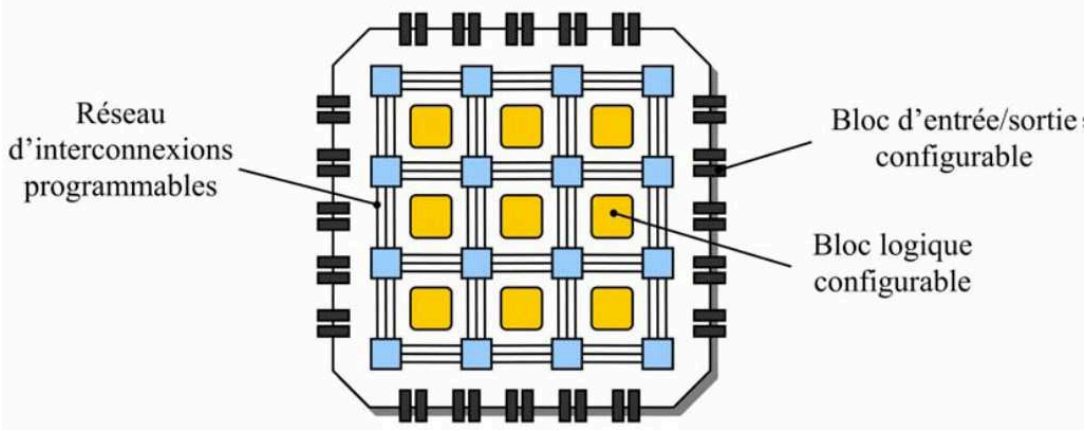


FIGURE 4.2: Schéma simplifié de l'architecture d'un FPGA.

de disposer de ressources d'interconnexion entre ces différentes cellules. Ce sont des bus qui remplissent cette fonction. Il existe différents types de bus d'interconnexion en fonction du type de signal à propager, comme le montre la Figure 4.3.

- Les interconnexions directes : permettent l'établissement des liaisons directes CLB-CLB et CLB-IOB.
- Les longues lignes : parcourent toute la longueur et la largeur du FPGA, elles permettent de transmettre les signaux entre les différents éléments, et d'éviter la multiplicité des points d'interconnexion.
- Les matrices d'interconnexion : permettent de relier un CLB à n'importe quel autre CLB sur le FPGA. Ce sont des aiguilleurs situés à chaque intersection. Elles raccordent les longues lignes entre elles selon diverses configurations. Chaque matrice d'interconnexion contient des buffers pour éviter l'affaiblissement des signaux traversant les longues lignes.

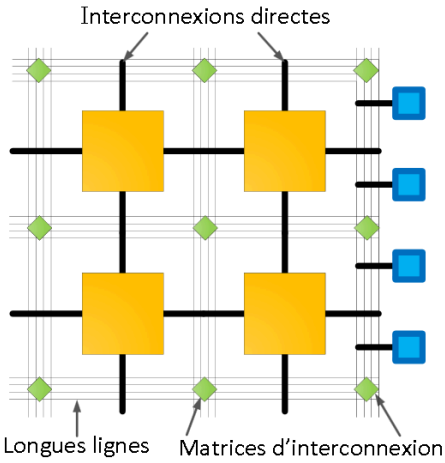


FIGURE 4.3: Interconnexion interne d'un FPGA.

En plus des trois blocs décrits dans le paragraphe précédent, les FPGA possèdent des fonctions pré-câblées comme des multiplicateurs et/ou des processeurs spécialisés implantés en matériel (Power-PC pour Xilinx, par exemple), au sein même des réseaux de portes logiques. Ces circuits permettent de spécialiser une architecture à moindre coût, par programme. Á ces ressources, les fabricants de FPGA ont aussi rajouté sur certains de leurs modèles, des blocs mémoire de $10 - 90\mathrm{Mbits}$, afin de stocker des données sans l'utilisation de mémoires externes.

L'un des principaux atout des composants FPGA est qu'ils sont reconfigurables à volonté, ce qui les rend parfait pour le prototypage. D'une part en raison de leur flexibilité, tout paradigme de calcul peut être implémenté sur FPGA. D'autre part, grâce à la reprogrammilité de ces circuits, il est possible de tester et déboguer sur du matériel réel. De plus, avec l'avancé de la technologie, les performances des FPGA se rapprochent de plus en plus de celles des Application Specific Integrated Circuit (ASIC) avec l'avantage de la reconfigurabilité.

L'un des principaux inconvénients des FPGA est le temps de développement, qui reste plus élevé que dans les approches purement logicielles. La programmation traditionnelle des FPGA est faite avec des langages de description de matériel (Hardware Description Language (HDL)), ce qui oblige l'utilisateur à concevoir à un niveau très bas. Les langages de haut niveau, tels que les extensions C, sont plus faciles pour les ingénieurs, mais le contrôle sur la conception est beaucoup plus faible [40].

4.3.3 *Configuration et reconfiguration des FPGA*

La configuration d'un circuit FPGA consiste à spécifier la fonctionnalité de chaque bloc logique et à organiser le réseau d'interconnexion afin de réaliser la fonction demandée, à l'aide d'un langage HDL, par exemple. Le code est ensuite synthétisé sous forme Register Transfer Logic (RTL). A partir de ce niveau, une synthèse physique est faite donnant un schéma de placement routage. Enfin, ce schéma est chargé dans la mémoire de configuration du FPGA, via un *bitstream*[1], configurant les CLB ainsi que les interconnexions entre ces CLB et les IOB (voir Figure 4.4).

L'architecture des FPGA a évolué ces dernières années, tout comme les ressources logiques configurables et la technologie de fabrication de ces circuits. Une des évolutions particulièrement intéressante concerne la reconfiguration des FPGA. Les nouvelles méthodes de reconfiguration *partielle* et *dynamique* ont ouvert un nouvel espace d'application pour ces circuits.

Lorsqu'une seule configuration est nécessaire au déroulement de l'application, on parle de reconfiguration *statique*. Les processus de reconfiguration et d'exécution de l'application sont séparés dans le temps. Une nouvelle reconfiguration sera effectuée dans deux cas :

- Une perte de la configuration initiale due pour certains dispositifs à un arrêt de la source d'alimentation.
- Une modification de l'application, si celle-ci est défaillante ou peut être améliorée.

1 bitstream : ensemble de bits permettant la configuration du FPGA

Une fois la configuration terminée, le FPGA fonctionne jusqu'à l'arrêt du composant ou au chargement d'une nouvelle reconfiguration.

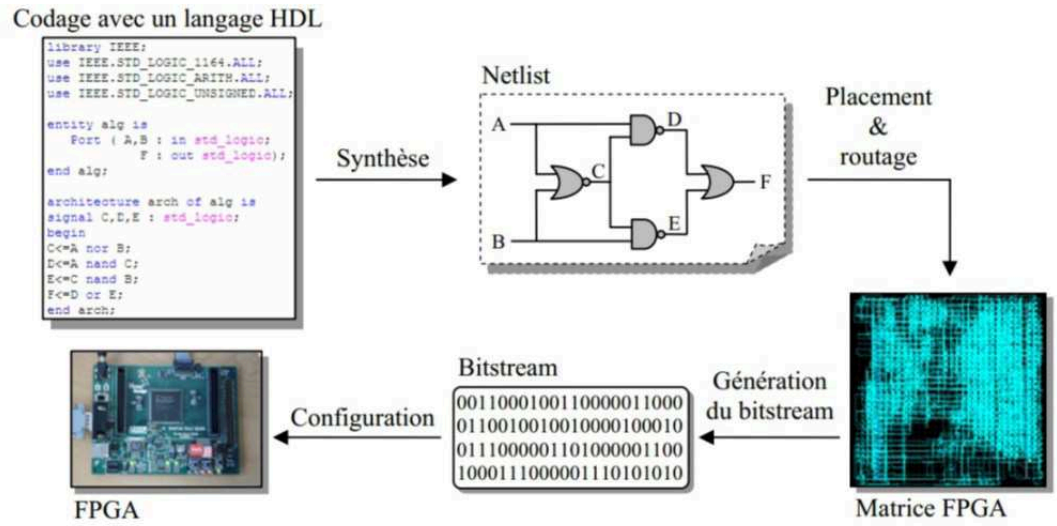


FIGURE 4.4: Flot de développement sur FPGA.

Contrairement à le reconfiguration statique, la reconfiguration dynamique peut se faire lors de l'exécution d'une application. Ce type de reconfiguration peut être décomposé en deux catégories : globale et partielle. Dans la reconfiguration dynamique globale, un ensemble de fichiers de configuration est mis en mémoire, ce qui permet d'appliquer une configuration totale de l'ensemble des ressources matérielles du FPGA à un instant t. Dans la reconfiguration dynamique partielle, une partie des éléments configurables est modifiée uniquement. Cette approche permet d'optimiser la surface configurée dans le temps et de réduire les temps de reconfiguration, tout en permettant au reste du système de garder son état de configuration initial.

La reconfiguration dynamique offre une flexibilité plus importante qu'avec une reconfiguration statique. En contrepartie, elle nécessite l'utilisation d'architecture à reconfiguration dynamique et des mécanismes plus complexes pour la gestion des bitstreams de reconfiguration. Plus de détails sur les techniques utilisées de reconfiguration dynamique sont présentés dans les travaux de Nicolas (2012) [141].

La Figure 4.5 présente schématiquement le déroulement des reconfigurations *statique* et *dynamique* durant l'exécution d'une application.

4.3.4 *Les nouvelles architectures SoC-FPGA*

Au fil du temps, les capacités et performances des FPGA ont augmenté de façon spectaculaire. Pour exemple, un FPGA moderne pourrait contenir des milliers d'additionneurs, de multiplicateurs, de Mbits de mémoires, ... Cependant, la problématique actuelle est que ces FPGA ne reflètent plus les capacités et les fonctionnalités des dispositifs programmables d'aujourd'hui.

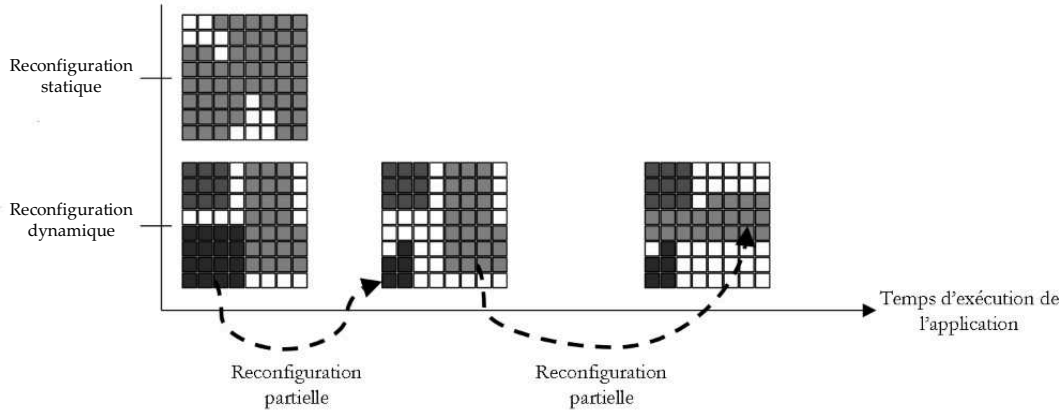


FIGURE 4.5: La reconfiguration statique et dynamique durant le temps d'exécution de l'application.

Pour cela, la technologie SoC-FPGA a vu le jour en englobant les outils et technologies d'aujourd'hui. Ce nouveau type de circuit intègre sur une seule puce une matrice FPGA et un processeur. À l'heure actuelle, il existe principalement trois SoC-FPGA sur le marché, comme le montre le Tableau 4.2. Les processeurs dans ces dispositifs sont entièrement dédiés. Chacune de ces trois familles utilisent un processeur Advanced RISC Machines (ARM) complet avec une hiérarchie complète de mémoire et des périphériques dédiés. En plus du processeur, un SoC-FPGA comprend un riche ensemble de périphériques, de mémoire sur puce, et un réseau de porte logique (FPGA).

La Figure 4.6 représente l'architecture du SoC-FPGA Cyclone-V de Altera, qui possède un processeur double cœur ARM Cortex-A9, un moteur de traitement multimédia NEON et une large gamme de périphériques standards.

Le processeur multi-coeur et la matrice FPGA sont alimentés séparément et peuvent être configurés et démarrés dans n'importe quel ordre. Une fois en fonctionnement, la partie FPGA peut être coupée pour réduire la consommation d'énergie du système. Le processeur multi-coeur ARM Cortex-A9 et le FPGA sont inter-connectés par des chemins de données à haut débit (plus de 125Gbps). Ce niveau des performances n'est pas possible avec deux puces. Un SoC-FPGA intégré permet aux concepteurs de cartes de supprimer les chemins d'E/S externes entre un processeur et un FPGA, réduisant ainsi la taille de la carte embarquée, la consommation d'énergie et le coût.

4.3.5 *Traitement d'images et FPGA*

De nombreuses algorithmes de traitement d'images (bas et moyen niveaux) sont parallèles et conviennent aux implémentations FPGA. Ces algorithmes n'utilisent pas d'informations haut niveau, et impliquent des opérations sur des pixels ou/et des régions. Le traitement peut se faire sur plusieurs éléments d'une même image simultanément. Le développement qu'ont connu les FPGA au cours des dernières années a conduit à un intérêt croissant pour leur utilisation comme plates-formes d'implémentation pour les applications de traitement d'images, en particulier en temps réel.

	ALTERA SOC FPGAS	XILINX ZYNQ-7000 EPP	MICROSEMI SMARTFUSION2
Processor	ARM Cortex-A9	ARM Cortex-A9	ARM Cortex-M3
Processor Class	Application processor	Application processor	Microcontroller
Single or Dual Core	Single or Dual	Dual	Single
Processor Max. Frequency	1.05 GHz	1.0 GHz	166 MHz
L1 Cache	Data : 32 KB Instruction : 32 KB	Data : 32 KB Instruction : 32 KB	No data cache Instruction : 8 KB
L2 Cache	Unified : 512 KB, with Error Checking and Correcting (ECC)	Unified : 512 KB	Not available
Memory Management Unit	Yes	Yes	Yes
Floating-Point Unit NEON Multimedia Engine	Yes	Yes	Not available
Acceleration Coherency Port	Yes	Yes	Not available
On-Chip Processor RAM	64 KB, with ECC	256 KB, no ECC	64 KB, no ECC
External Memory Controller	Yes	Yes	Yes
Memory Types Supported	LPDDR2, DDR2, DDR3L, DDR3	LPDDR2, DDR2, DDR3L, DDR3	LPDDR, DDR2, DDR3
External Memory Bus Max. Frequency	400 MHz (Cyclone V), 533 MHz (Arria V)	533 MHz	333 MHz
FPGA Fabric	Cyclone V, Arria V	Artix-7, Kintex-7	Fusion2
FPGA Logic Density Range	25K to 462K	28K to 444K	6K to 146
Boot Sequence	Processor first, FPGA first, or both simultaneous	Processor first	Processor boot, FPGA non-volatile

TABLE 4.2: Caractéristiques des SoC-FPGA disponible dans le commerce.

Le parallélisme des algorithmes de traitement d'images existe sous deux formes majeures : un parallélisme spatial et un parallélisme temporel. Une implémentation FPGA peut utiliser les deux formes de parallélisme séparément ou un mélange des deux. Par exemple, afin d'exploiter ces deux formes, le FPGA peut être configuré pour partitionner l'image et distribuer les parties résultantes sur plusieurs unités de traitement qui pourraient toutes traiter les données en même temps (un traitement Single Instruction Multiple Data (SIMD)). Si les unités de traitement sont différentes et exécutent un processus complémentaire on parle d'un traitement *pipeline*.

Dans la pratique, ces deux formes de parallélisation sont soumises aux contraintes du mode de traitement et du matériel choisi. Il existe trois modes de traitement : en-ligne (sur le flux), hors-ligne et le traitement hybride. Dans le premier mode, les données doivent être traitées à une fréquence (vitesse) supérieure ou égale à celle du système d'acquisition de données, c'est la contrainte du *temps de traitement*. Dans le mode hors-ligne, les données à traiter sont mises dans une mémoire, donc la vitesse d'exécution dans la plupart des cas est limitée par la vitesse d'accès mémoire. Le mode hybride est un mélange des deux traitements en-ligne et hors-ligne. Dans ce cas, la contrainte temporelle est détendue afin que l'image soit capturée à un rythme plus lent.

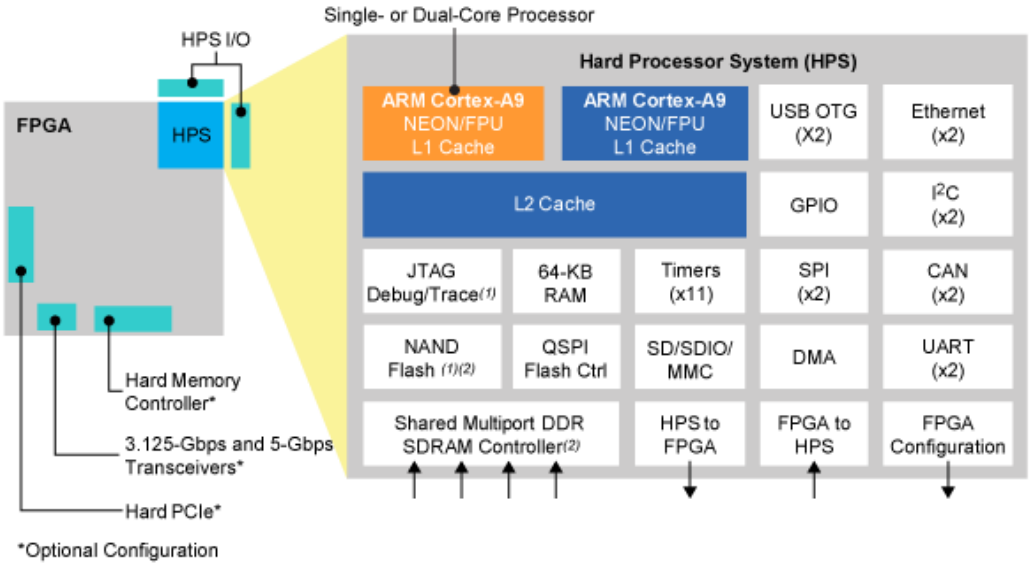


FIGURE 4.6: Architecture du SoC-FPGA Cyclone-V.

Lorsque on utilise un traitement sur le flux, certaines opérations ont besoin pour faire un calcul particulier d'un voisinage (comme par exemple le filtrage Gaussien, Median). Pour cela l'image doit être partiellement ou totalement mise en mémoire tampon. Par conséquent de nouvelle contraintes viennent s'ajouter lors de l'implémentation d'un algorithme de traitement d'image sur FPGA : *les contraintes de ressources* et *de bande passante*.

La taille de la mémoire tampon à utiliser dépend du traitement que l'on veut appliquer à l'image. Dans le pire des cas (Homographie, par exemple) toute l'image doit être mise en mémoire. Une image couleur 24bits (8bits par canal de couleur) avec une taille 640×480 pixels nécessite 7.0Mbits de mémoire. Les FPGA ont des quantités très limitées de RAM[2] embarquée sur puce. Par exemple le Cyclone-III ne contient que 3.8Mbits de mémoire. Les blocs logiques eux-mêmes peuvent être configurés pour être utilisés comme des blocs mémoires, mais c'est généralement une solution à éviter pour des raisons évidentes d'encombrement et de vitesse.

Pour résoudre ce problème de une mémoire externe est utilisée. Cette solution ne permet qu'un seul accès à la mémoire par cycle d'horloge, ce qui peut être un problème pour les nombreuses opérations qui nécessitent un accès simultané à plus d'un pixel de l'image. Par exemple, l'interpolation bilinéaire requiert un accès simultané à quatre pixels de l'image. Les alternatives possibles pour faire face à ce problème sont : plusieurs bancs de mémoires en parallèle, une mémoire à accès multiple ou une mémoire ayant une vitesse de lecture plus rapide que la vitesse de notre traitement. Ces solutions ont toutes des inconvénients liés soit au coût, l'espace et/ou la synchronisation. La contrainte de ressources se pose en raison du nombre limité des ressources disponibles dans le FPGA tels que mémoire interne/externe ou d'autres blocs fonctionnels (CLB, IOB, DSP, ...).

2 Random Access Memory (RAM)

Lorsqu'on suit la chaîne de traitement d'images complète du bas jusqu'au haut niveau, le parallélisme exploitable dans les différents traitements diminue car on passe d'une représentation pixellique de l'environnement à une représentation descriptive. Cela réduit la quantité de données qui doit être traitée, ce qui permet de gagner plus d'espace de stockage et de temps de traitement. Les différents algorithmes de traitement d'images (bas et moyen niveau) peuvent être classés comme suit :

4.3.5.1 *Opérations pixelliques (ponctuelles)*

Dans cette classe d'opérations la valeur de chaque pixel de sortie ne dépend que de la valeur du pixel d'entrée lui correspondant. L'implémentation de ces opérations ponctuelles sur du matériel peut être réalisé simplement en faisant passer l'image, un pixel à la fois dans un bloc de traitement ; conçu pour effectuer l'opération de point requis.

Ces opérations sont mutuellement indépendantes et peuvent donc être facilement parallélisées ou introduites dans un pipeline. Un traitement pixellique est aussi possible sur plusieurs images en même temps ($I_(output) = f(I_1, I_2, \ldots)$), cependant l'accès à ces différentes images nécessite une gestion soigneuse de la mémoire. Des exemples de ces fonctions sont des opérations arithmétiques et logiques, comme par exemple l'addition, la multiplication et la comparaison des pixels.

Les opérateurs phalliques peuvent être classées en deux catégories :

- Opération pixellique indépendante des données : dans cette catégorie un seul passage sur l'image est suffisant pour avoir le résultat souhaité. La valeur d'un pixel de sortie n'est déterminée qu'à partir de la valeur du pixel d'entrée lui correspondant. Les exemples incluent : binarisation d'une image, conversion d'espace de couleur, correction de luminosité, opérations arithmétiques et opérations logiques.
- Opération pixellique dépendante des données : la valeur d'un pixel de sortie est déterminée à partir de la valeur du pixel d'entrée lui correspondant, comme les méthodes mentionnées ci-dessus, mais la transformation dépend de statistiques extraites de l'image source. Dans ce cas-la plusieurs passes sur l'image source sont nécessaires, au moins deux fois. Les exemples incluent : l'égalisation d'histogramme et la transformée de Hough.

Le principal inconvénient de l'implémentation de ces opérations sur des systèmes embarqués, est que des contraintes au niveau mémoire peuvent apparaître sur certaines les plates-formes matérielles.

4.3.5.2 *Opérations sur fenêtre (voisinage)*

Ces opérations calculent la valeur d'un pixel de l'image de sortie en fonction du voisinage du pixel lui correspondant dans l'image d'entrée. Cela se fait concrètement par le glissement d'une fenêtre sur l'image d'entrée (voir Figure 4.7). Les différents traitements de filtrage d'image appartiennent à cette catégorie d'opérations. L'approche directe pour réaliser ce type de traitements consiste à stocker toute l'image en mémoire, puis à accéder aux pixels nécessaires pour le traitement afin de produire l'image de sortie.

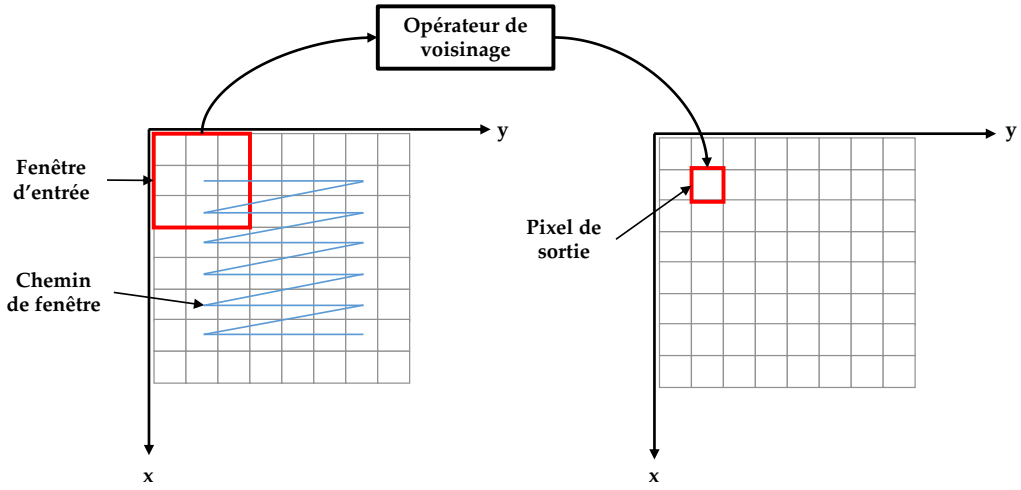


FIGURE 4.7: Exemple conceptuel de filtrage par fenêtre glissante.

Si on désire un traitement sur un voisinage ($n \times n$) en temps réel sur le flux de vidéo, alors pour chaque pixel en sortie $n \times n$ pixels sont nécessaires pour effectuer les calculs à chaque fois que la fenêtre est déplacée. Contrairement aux opérateurs pixellique, le pixel entrant ne peut être traité lorsqu'il est reçu ; le calcul commence lorsque tous les pixels de son voisinage sont disponibles. Comme les pixels sont habituellement fournis/lus suivant un balayage horizontale de l'image, alors selon la taille du masque (voisinage) nécessaire pour le calcul, plusieurs lignes de l'image traitée doivent être mise en mémoire. En considérant une image de taille $L \times C$ et un masque carré de taille $n \times n$, alors il faut stocker dans la mémoire au moins $N = C * (n-1) + n$ pixels. Le temps de latence minimale du système est de N/f, où f est la fréquence d'horloge du système. Les exemples incluent : la morphologie mathématique, la convolution, le filtrage et le traitement pyramidale ...

L'implémentation matérielle du bloc permettant de garder en mémoire tampon les $N = C * (n-1) + n$ pixels se fait par des registres à décalage (ou mémoires tampon circulaire). La Figure 4.8 montre le principe de ce bloc pour une fenêtre de traitement de taille 3×3.

Dans ce mode de traitement les problèmes de mémoire peuvent survenir si de grands masques sont utilisés. Pour le filtrage la plus petite taille de masque adoptée est généralement de 3×3 pixels, afin de surmonter les limitations mémoire. Le degré de parallélisme qui peut être déployé dans les opérations sur fenêtre, dépend de la mémoire connectée aux éventuelles processus concurrents.

4.3.5.3 *Opérations globales*

Additionner toutes les valeurs de pixel de l'image, compter les pixels égale à zéro, trouver la moyenne, l'écart-type, les valeurs minimales et maximales sont des opérations qui nécessitent un seul passage sur les pixels d'une image. Par rapport aux opérations

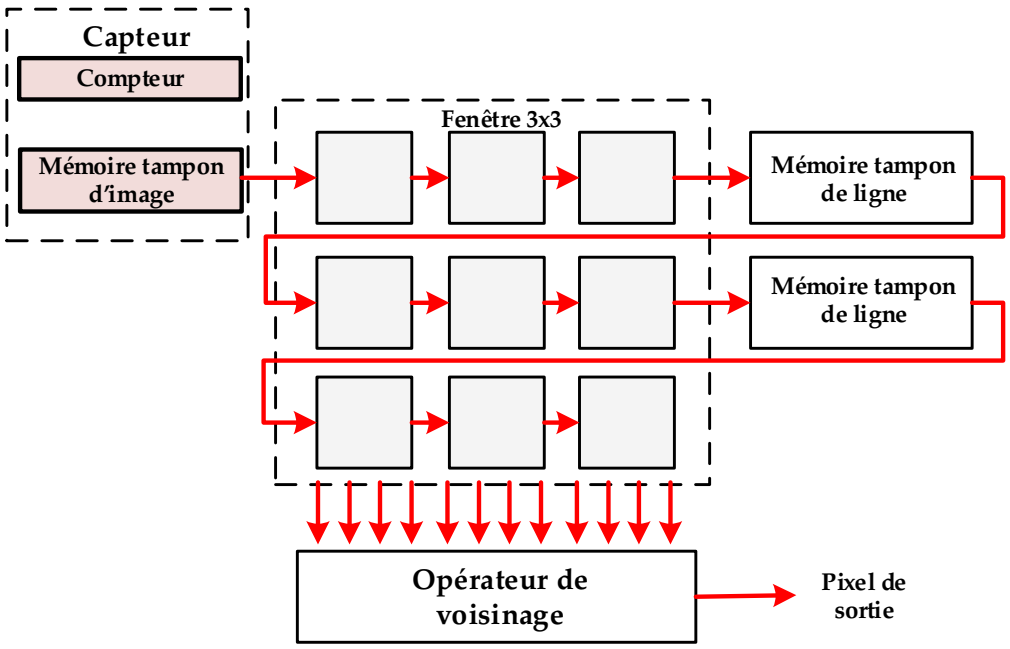


FIGURE 4.8: Architecture de mise en œuvre d'un convolueur sur flot.

précédentes dans les opérations globales le résultat final dépend de tous les pixels de l'image qui doivent être stockées en mémoire, avant tout calcul.

Dans ce mode de traitement le principal inconvénient est la taille d'image à traiter, qui doit être mise en mémoire. En raison de la limitation des ressources internes des plates-formes reconfigurables (en mémoire), les opérations globales ne sont pas forcément bien adaptées à des implémentations matérielles. Si les resources internes de la plate-forme matérielle utilisée sont insuffisantes, l'utilisation de mémoire externe est inévitable. Parmi les algorithmes appartenant à cette catégorie : la rotation d'image et la segmentation d'image ...

Pour terminer cette section et afin de montrer l'avantage des FPGA par rapport aux autres plates-formes de calcul, nous présentons l'étude faite par Nieto et al. (2012) [142], dans laquelle les auteurs comparent les performances obtenues lorsque l'algorithme SURF est implémenté sur un Central Processing Unit (CPU), GPU et un FPGA, (voir Tableau 4.4)

AUTEUR(S)	PLATE-FORME	FRÉQUENCE D'IMAGE	PUISSANCE (W)
Bouris et al. (2010) [22]	Intel Core 2 Duo 2.4 GHz	< 7 fps	65
Zhang (2010) [203]	Intel Core 2 Duo P8600 2.4 GHz	33 fps	45
Terriberry et al. (2008) [175]	nVidia GeForce 880 GTX	56 fps	200
Bouris et al. (2010) [22]	Xilinx Virtex 5XC5VFX130T	70 fps	< 20

TABLE 4.3: Les différentes implémentations de SURF sur différentes plates-formes pour des images de 640×640 pixels.

Comme on peut le voir sur le tableau ci-dessus, l'implémentation à base de FPGA offre les meilleures performances en termes de vitesse et de consommation d'énergie.

4.4 ÉTAT DE L'ART DES ARCHITECTURE DE TRAITEMENT D'IMAGE EXISTANTES

Dans ce qui suit nous présentons des exemples d'architectures à base de FPGA et autres plates-formes de calculs, qui sont conçues pour des applications de traitement d'images. Les premiers travaux trouvés dans la littérature ne comprenaient que les algorithmes de *calculs lourds*, mais simples comme la conversion d'espace de couleur [20], la soustraction de fond et la détection de primitives [7, 159].

Moorhead and Binnie (1999) [131] ont présenté dans leur article une des premières plates-formes de calcul de traitement d'image en CMOS, pour la détection de contour. Une autre plate-forme utilisé pour la même application, était proposé par Albani et al. (2002) [5]. Cette architecture nommée VISoc, est composée d'un capteur CMOS 320×256 pixels et d'un processeur RISC 32 bits. Arth et al. (2006) [8] ont proposé la TRICam, une plate-forme à base de DSP utilisée principalement pour la détection d'objets. Bauer et al. (2007) [13] présentent dans leur article une plate-forme à base d'un DSP, cette dernière détecte le changement d'intensité dans une scène ce qui permet de détecter les objects en mouvement et d'estimer leurs vitesses.

Toutefois dans la suite de ce chapitre, nous nous focalisons sur les architectures de traitement d'images embarqué de type ***caméra intelligente*** ou ***smart camera***. Ces plates-formes offrent la possibilité d'extraire des primitives ou des objets de la scène observée au sein du système d'acquisition en réduisant la quantité de données à transmettre aux étapes de traitement supérieur (un PC hôte, par exemple). La différence entre une caméra intelligente et une caméra traditionnelle est la nature des tâches effectuées par le processeur d'image intégré et le résultat généré par la caméra en sortie. La Figure 4.9 présente la structure générale d'une caméra intelligente. Dans ce mémoire, nous définissons une caméra intelligente comme étant un système de vision embarqué permettant de produire non pas un simple flux vidéo mais des informations ayant un certain niveau de sémantique.

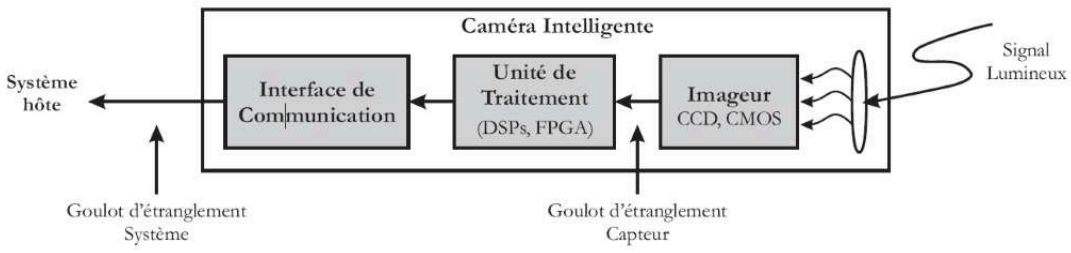
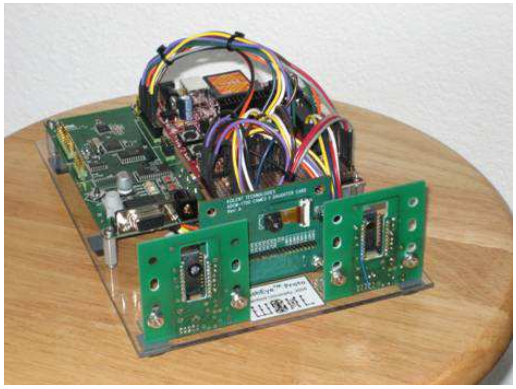


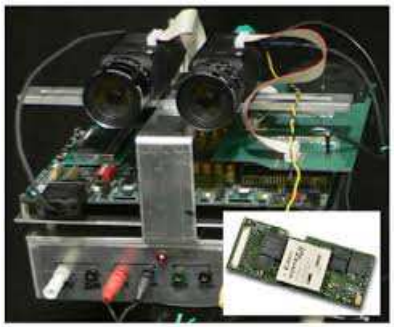
FIGURE 4.9: Structure générale d'une caméra intelligente.

Wolf et al. (2002) [194] ont développé la première génération de caméra intelligente pour la reconnaissance des gestes en temps réel. Cette architecture était composée d'une carte Peripheral Component Interconnect (PCI) comportant deux processeurs Trimedia-TM1300 intégrée dans un PC hôte et d'une camera video Hi8. Une autre architecture a été proposée par Hengstler et al. (2007) [73] à l'Université de Stanford, nommée Me-

shEye (voir Figure 4.10a) pour de la surveillance vidéo. Ce système multi-capteurs à résolution hybride permet l'optimisation du processus d'acquisition. Composé d'un capteur basse résolution utilisé pour la détection de mouvements d'objets dans une scène, et d'un capteur haute résolution utilisé pour l'acquisition d'une fenêtre d'intérêt contenant cet objet.



(a) Architecture du système de vision MeshEye.

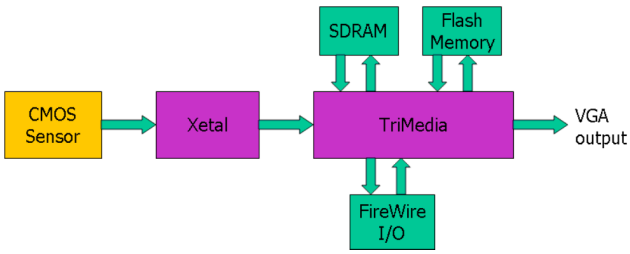


(b) Architecture du système de vision StéreoCam.

FIGURE 4.10: Architectures de traitement d'image.

L'INCA+ est un exemple de caméra intelligente (Figure 4.11a), composée d'un capteur CMOS et de deux processeurs (Xetal et TriMedia). Le premier processeur Xetal est adapté pour le parallélisme inhérent des opérations de traitement d'images bas niveau, tandis que le processeur TriMedia est utilisé pour exploiter le parallélisme au niveau des instructions, ce qui permet d'exécuter plusieurs opérations indépendantes par cycle. La Figure 4.11b représente le schéma synoptique de l'architecture de l'INCA+.

(a) La caméra INCA+.



(b) Schéma synoptique de l'architecture de l'INCA+.

FIGURE 4.11: La caméra intelligente INCA+.

On peut aussi trouver dans la littérature des systèmes de vision intégrant un FPGA en tant qu'unité de traitement, ce qui permet d'exécuter un large éventail d'algorithmes de traitement d'images de manière optimale. Van Der Horst et al. (2006) [186] proposent un système de vision où deux caméras DICA321 et un FPGA de Xilinx sont utilisées pour constituer une plate forme stéréo-vision (voir Figure 4.10b). Cette architecture est conçue comme un système généraliste de vision et traitement d'images, cependant dans l'article [186] elle est utilisé pour l'estimation de profondeur dans une scène par le calcul de disparité qu'il y'a entre deux images.

Le laboratoire Le2i a développé un système de vision (voir Figure 4.12) à base de FPGA [133]. Cette plate-forme est composée d'un capteur CMOS haute-vitesse (1.3Mpixels), d'un circuit FPGA de la famille Virtex II de chez Xilinx et d'une interface USB-2.0 [3]. Différents algorithmes de traitement d'images ont été implémentées sur cette architecture, comme la compression d'image, le filtre de Sobel ou des opérations d'érosion/dilatation.

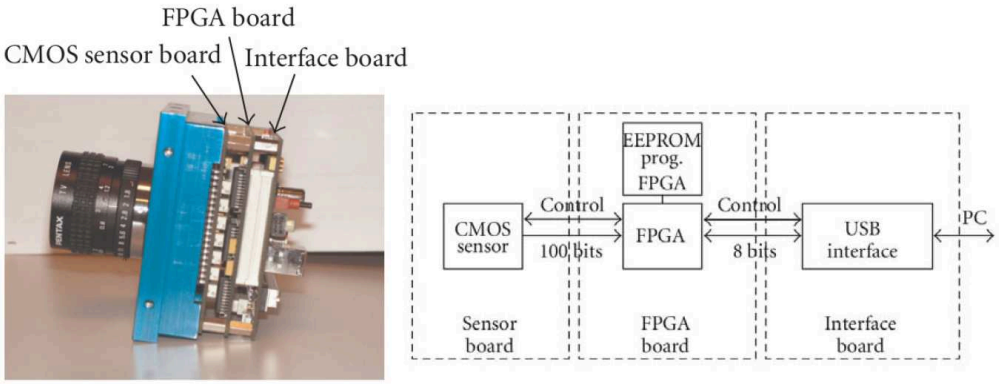


FIGURE 4.12: Caméra rapide du laboratoire Le2i.

Une autre architecture développée au sein de l'Institut Pascal à base de FPGA est décrite dans l'article de Dias et al. (2007) [45]. Le FPGA est utilisé pour l'implémentation de différents modules afin de contrôler l'imageur CMOS, la transmission de données (via le Firewire) et les mémoires externes, ainsi que pour l'implémentation d'algorithmes de traitement d'image. Pelissier and Berry (2011) [156] ont développé une caméra intelligente dédiée aux traitements stéréo, nommée BiSeeMOS (voir Figure 4.13). Cette caméra est composée de deux capteurs d'image CMOS, d'un FPGA Altera Cyclone-III EP3C120 et d'une carte de communication USB2.0. Une des applications implémentée sur cette caméra est la transformée de Census d'une image. Sur cette plate-forme des images de résolution 1024×1024 à une cadence de 100 images par seconde.

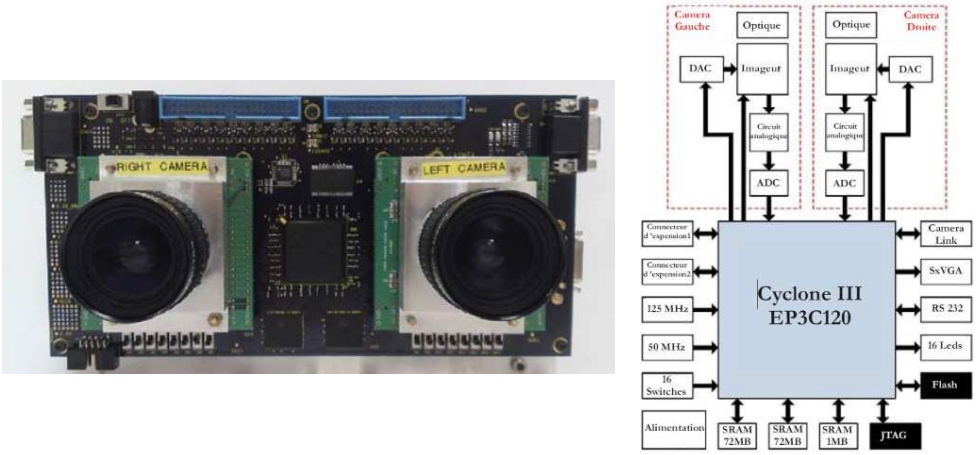


FIGURE 4.13: Caméra intelligente BiSeeMOS.

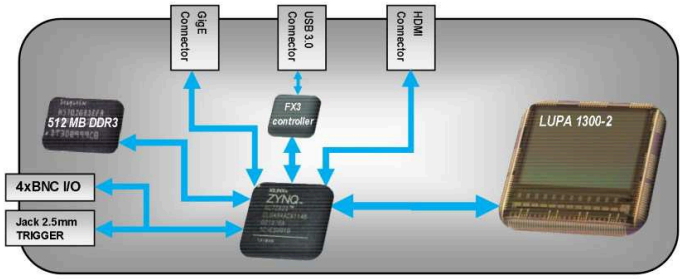
3 Universal Serial Bus (USB)

L'entreprise *SEE FAST TECHNOLOGIES* propose actuellement deux produits de type caméras rapides intelligentes à base de FPGA :

- **ProcImage250** : est une version améliorée de la ProcImage240 (voir Figure 4.14-a). La carte mère est composée d'un FPGA Spartan 3, d'une mémoire SRAM[4] rapide et d'une liaison USB2.0. Différentes applications sont implémentées sur cette caméra comme par exemple : détection de défauts, asservissement dynamique à des phénomènes rapides, et rééducation.
- **ProcImage500-Eagle** : est une caméra rapide dédiée aux traitements embarqués en temps réel d'images prises à hautes cadences (voir Figure 4.14-b). La carte mère est composée d'un FPGA Zynq de Xilinx, d'une mémoire DDR3 et d'une liaison USB3.0, la Figure 4.14-c représente l'architecture de cette caméra. Elle permet de réaliser des traitements d'images embarqués ultra rapides et exploitables en temps réel, comme par exemple : le suivi de points d'intérêt et la reconnaissance de formes.

(a) ProcImage250.

(b) ProcImage500-Eagle.



(c) Architecture interne de la caméra ProcImage500-Eagle.

FIGURE 4.14: Architecture interne de la caméra ProcImage500-Eagle.

D'autres entreprises ont proposé aussi des caméras intelligentes à base de FPGA, comme par exemple : *Eye Vision Technology (EVT)*, *OptoMotive*, *FastVision* et *Tattile* ... La Figure 4.15 représente les caméras intelligents de ces différentes entreprises.

Le Tableau 4.4 présentes un aperçu des plates-formes de type caméras intelligentes, trouvées dans la littérature depuis 1999 jusqu'à présent.

SYSTÈMES	PLATES-FORMES				APPLICATIONS
	CAPTEUR	CPU	COMM	ÉNERGIE	
Moorhead and Binnie (1999) [131]	Capteur CMOS	Logique personnalisée sur puce	-	Secteur	Détection de contours bas niveau
VISoc Albani et al. (2002) [5]	Capteur CMOS, 320 × 256	Processeur RISC 32bits	-	Batterie	Détection de contours bas niveau
Wolf et al. (2002) [194]	Caméra video Hi8	Processeurs Trimedia TM1300	-	Secteur	Reconnaissance des gestes
Leeser et al. (2004) [107]	Caméra Dalsa 1M30P	FPGA Xilinx Virtex 2000E	-	-	Domaine médical

Suite page suivante ...

4 Static Random Access Memory (SRAM)

(a) RazerCam (EVT). (b) The FastCamera 300 (FastVision). (c) The FastCamera 405 (FastVision)

(d) ActuaTech. (e) Cameleon (OptoMotive).

FIGURE 4.15: Caméras intelligentes qu'on trouve sur le marché.

SYSTÈMES	PLATES-FORMES				APPLICATIONS
	CAPTEUR	CPU	COMM	ÉNERGIE	
Dickinson et al. (2005) [46]	Caméra	FPGA Xilinx Virtex II	-		Système de surveillance du nourrisson
StéreoCam Van Der Horst et al. (2006) [186]	Caméras DICA321	FPGA Xilinx V2Pro7	FireWire	-	Estimation de profondeur
TRICam Arth et al. (2006) [8]	Video (pas de capteur)	DSP	Ethernet	Secteur	Détection d'objet
INCA+ Fatemi (2007) [51]	Capteur CMOS	Processeurs Xetal Trimedia	FireWire	Batterie	Détection et reconnaissance de visage
MeshEye Hengstler et al. (2007) [73]	2× low resolution sensor, 1× VGA color CMOS sensor	ARM7 @ 55MHz	802.15.4	Batterie	La surveillance par détection d'objet
Bauer et al. (2007) [13]	Capteur CMOS, 64 × 64	Blackfin DSP	-	Secteur	Détection de véhicule et estimation de vitesse
Caméra Le2i Mosqueron et al. (2007) [133]	Capteur CMOS, 1.3 Mpixels	FPGA Xilinx Virtex II	USB 2.0	-	Compression d'images - Filtre de Sobel
Dias et al. (2007) [45]	Capteur CMOS, 2048 × 2048	FPGA Altera Stratix	Firewire	Secteur	Suivi d'objets
EyeBot-M6 Dietrich (2009) [48]	Two color cameras, 352 × 288	FPGA Xilinx Spartan3E500	USB 2.0	-	Détecteur de points d'intérêt
HPSC Johannes Furtler and Eckel (2010) [89]	Capteur Aptina MT9M413	FPGA Altera 2S60	-	-	Systèmes de contrôle de qualité

Suite page suivante ...

SYSTÈMES	PLATES-FORMES				APPLICATIONS
	CAPTEUR	CPU	COMM	ÉNERGIE	
Norouznezhad et al. (2010) [143]	Capteur Micron MT9P001	FPGA Xilinx Virtex-5	-	Secteur	Suivi d'objet
Diederichs (2011) [47]	Camera 2592 × 1944 pixels	FPGA Xilinx Spartan-3	-	-	asservissement visuel
BiSeeMOS Pelissier and Berry (2011) [156]	Deux capteur CMOS	FPGA Altera Cyclone	USB 2.0	Secteur	Transformée de Census
Bravo et al. (2011) [23]	Capteur CMOS 1.2 Megapixel	FPGA Xilinx XC4VFX12	Ethernet	-	-
NIR camera Zeng (2012) [201]	Capteur CMOS NIR	FPGA Xilinx Spartan-6	-	-	la détection de glace/eau sur la surface de la route
HDR-ARtiSt Lapray et al. (2013) [103]	Capteur CMOS, 1.3 Mpixels	FPGA Xilinx Virtex 6	-	Secteur	Vidéo HDR
ProcImage250 Technologies (2014) [173]	Capteur CMOS, 640 × 480	FPGA Xilinx Spartan 3	USB 2.0	-	Vidéo rapide
ProcImage500-Eagle Technologies (2014) [174]	Capteur CMOS, 1280 × 1024	FPGA Xilinx Zynq	USB 3.0	-	Différent traitement d'image
DreamCam Merwan Birem (2014) [123]	Capteur CMOS, 1280 × 1024	FPGA Altera Cyclone	USB 2.0	Secteur	Extraction de primitives

TABLE 4.4: Caméra intelligente.

4.5 CONCLUSION

Dans ce chapitre nous avons présenté les trois chaînes de traitement d'images bas, moyen et haut niveau. La structure des FPGA a été introduite en tant que composant pour le calcul intensif de bas et moyen niveau du traitement d'image. Actuellement, avec l'augmentation de la taille des FPGA et l'apparition des SoC-FPGA, il est possible de mettre en œuvre des systèmes complets ayant les trois niveaux de traitement d'image sur une seule puce.

Dans le chapitre suivant l'architecture du capteur de type caméra intelligente nommé *DreamCam* est présentée.

5

LA DREAMCAM

5.1 INTRODUCTION

Dans la première partie de ce chapitre, nous donnons une présentation générale de l'architecture de la caméra intelligente « *DreamCam* » utilisée au cours de nos travaux de recherche. Cette caméra a été développée au sein de l'Institut Pascal (Ex : Laboratoire des Sciences et Matériaux pour l'Électronique, et d'Automatique (LASMEA)). Lors de la conception de la *DreamCam* le choix s'est porté sur l'utilisation d'un FPGA, car comme cela a été expliqué précédemment ces composants sont idéaux pour les implémentations d'algorithmes de traitement d'images [200].

Dans la seconde partie de ce chapitre et afin de montrer l'efficacité de la *DreamCam*, nous avons choisi d'implémenter deux algorithmes de traitements d'image : le filtre Sobel pour la détection de contour et l'algorithme de détection de points d'intérêt Harris & Stephen.

5.2 DESCRIPTION MATÉRIELLE DE « DREAMCAM »

La *DreamCam* est une caméra intelligente entièrement personnalisable basée sur un imageur de type CMOS. La principale originalité de ce capteur est le fait qu'il interfacé par un système sur puce (SoC) intégré dans un FPGA. L'approche présentée permet d'exécuter dans l'unité principale de traitement (le FPGA) la plupart des procédés de perception et de traitement d'images, ce qui permet de réduire le goulot d'étranglement de communication classique, en envoyant, uniquement les descripteurs de primitives détectées et non pas toute l'image. Un autre avantage de cette approche est la rétroaction en temps réel sur l'imageur, c'est-à-dire, la possibilité de régler activement les paramètres de l'imageur afin d'optimiser la perception.

La plupart des applications de vision se concentrent sur de petites zones de l'image et par conséquent, l'acquisition de la totalité de l'image n'est pas toujours nécessaire. Il est donc évident que l'un des principaux objectifs, d'un capteur de vision efficace, est de pouvoir acquérir des régions d'intérêts (Region Of Interest (ROI)) dans l'image et de concentrer les ressources de traitement sur ces régions uniquement. La notion d'étude locale est alors prédominante et le choix de la technologie d'imagerie devient crucial. Cela explique pourquoi l'imageur CMOS a été choisi. Il est généralement admis que la technologie CMOS remplacera la technologie CCD classique dans de nombreuses applications, en raison de ses capacités et cela :

- Pour permettre l'accès à des régions spécifiques de l'image (ROI),
- Pour permettre des vitesses d'acquisition plus élevées,

- Pour permettre la fonctionnalité sur puce (caméra sur puce ou rétine),
- Pour fournir une gamme beaucoup plus dynamique (allant jusqu'à 120dB),
- Pour être basée sur les processus de fabrication standard.

5.2.1 *Architecture matérielle globale*

L'architecture de la *DreamCam* est composée de cinq cartes reliées entre elles (voir la Figure 5.11). Le cœur de ce système est un FPGA qui permet une grande polyvalence. La structure modulaire de cette caméra donne la possibilité de mettre à jour où remplacer facilement les différentes cartes, afin de modifier le type d'imageur ou la carte de communication par exemple. Cette caméra peut fonctionner avec deux imageurs différents et peut aussi utiliser soit un USB-2.0 pour la transmission de données, soit un moyen de communication de type Giga-Ethernet. Chaque carte est décrite en détail dans ce qui suit.

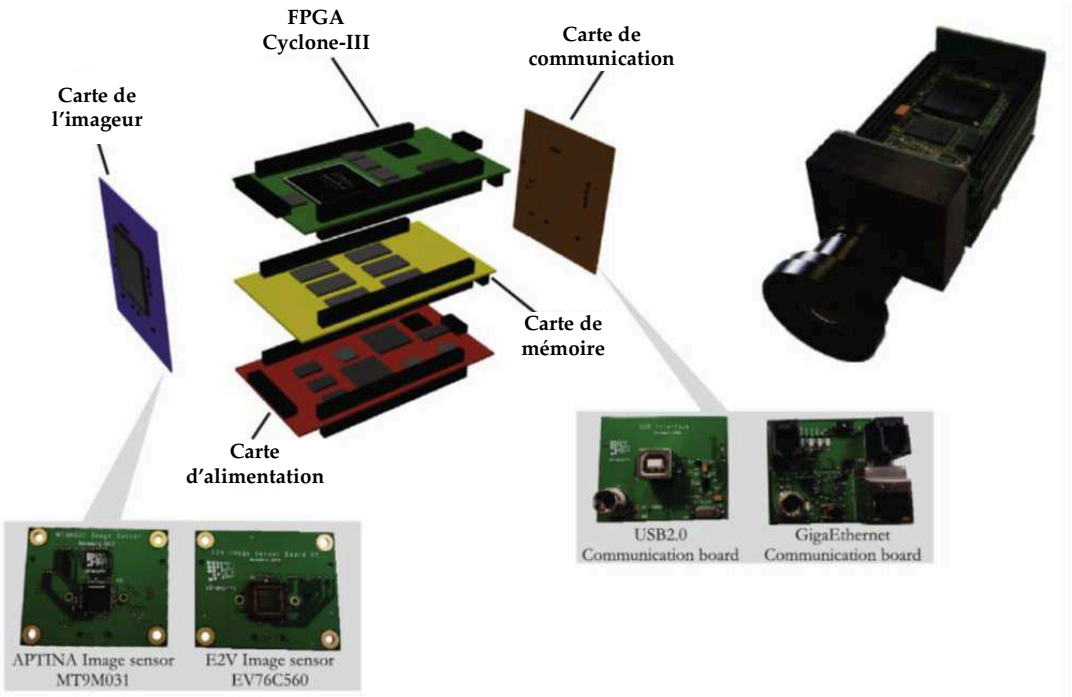


FIGURE 5.1: Vue d'ensemble de la *DreamCam*.

CARTE D'IMAGEUR : Les deux cartes d'imageurs développées sont basées sur une architecture électronique similaire. Les imageurs utilisés dans ce travail sont les suivants :

- L'imageur MT9M031 : Ce capteur d'image CMOS est fabriqué par Aptina[1]. Il peut fonctionner à 45 images/s en pleine résolution de 1280 × 960 pixels ou à la vitesse de 60 images/s à une résolution de 720pHD. La consommation

1 **Aptina Imaging Corporation** : est une société qui vend des produits d'imagerie CMOS

d'énergie est de 270m*W* en mode 720p60. La gamme dynamique est égale à 83.5dB, assez grande pour un capteur basé sur le mode global shutter,

- L'imageur EV76C560 : Il s'agit d'un imageur de 1.3M pixel (1280×1024) capteur d'image CMOS dédié à la vision industrielle qui peut fonctionner en mode global ou rolling shutter[2]. La conception des pixels offre d'excellentes performances dans des conditions de faible éclairage avec une vitesse de 60 images/s en pleine résolution. Le nouveau mode d'intégration, de lecture des pixels et de pré-traitement d'image embarquées sur cet imageur ont permis l'obtention de meilleures performances.

CARTE DE TRAITEMENT : C'est la partie principale du système. Le cœur est un FPGA de type Cyclone-III EP3C120 de la société Altera (voir Figure 5.12). Les principales caractéristiques de ce FPGA utilisé sont détaillées dans le Tableau 5.1. La nécessité d'une forte parallélisation a conduit à relier $6 \times 1MWords$ blocs mémoire asynchrone SRAM à ce FPGA. Chaque mémoire a un bus de données et un bus d'adresse privé. Par conséquent, six processus (utilisant 1M*W* chacun) peuvent accéder à toutes les mémoires en même temps. La famille de FPGA à faible puissance Cyclone-III a été choisie pour les raisons suivantes :

- Tout d'abord, son architecture se compose de 120K éléments logiques disposés verticalement (Logic Element (LE)s), $4Mbits$ de mémoires embarquées disposées en $9kbits$ blocs et 288 multiplicateurs embarqués 18×18.

- Deuxièmement, le Cyclone intègre des DSP blocs. Ces blocs embarqués ont été optimisés pour implémenter plusieurs fonctions DSP offrant des performances maximales et une utilisation minimale des ressources logiques. En outre, ces blocs DSP intégrés peuvent être utilisés pour créer des algorithmes DSP et des routines mathématiques complexes de haute performance.

- Enfin, le Cyclone est optimisé pour maximiser les performances d'une intégration SoC basée sur le processeur NIOS[3] embarqué.

CARTE DE COMMUNICATION : Cette carte est connectée à la carte mère et gère toutes les communications avec l'ordinateur hôte. Actuellement, la carte de communication peut être soit en USB-2.0 ou en Giga-Ethernet.

- USB-2.0 est géré par le micro-contrôleur Cypress CY7C68013. Il intègre un processeur amélioré basé sur un noyau-8051 et le jeu d'instructions est compatible avec la norme 8051.

- Le protocole Giga-Ethernet est pris en charge par l'émetteur-récepteur Marvel 88E1111. Il s'agit d'un dispositif de couche physique comprenant un seul Gigabit Ethernet (GbE) émetteur-récepteur.

CARTE DES MÉMOIRES : La banque de mémoires contient six mémoires asynchrones SRAM, dont chacune a une taille de $1MWords$. Ces mémoires sont de $16Mbits$

2 Le **rolling shutter** (littéralement "obturateur déroulant") est une technique photographique où l'on enregistre une image en déplaçant l'obturateur. L'obturateur peut être de nature mécanique ou électronique.

3 Le NIOS est un Processeur softcore propriétaire de Altera. Il est basé sur un cœur RISC $32bits$. Il est doté du bus Avalon.

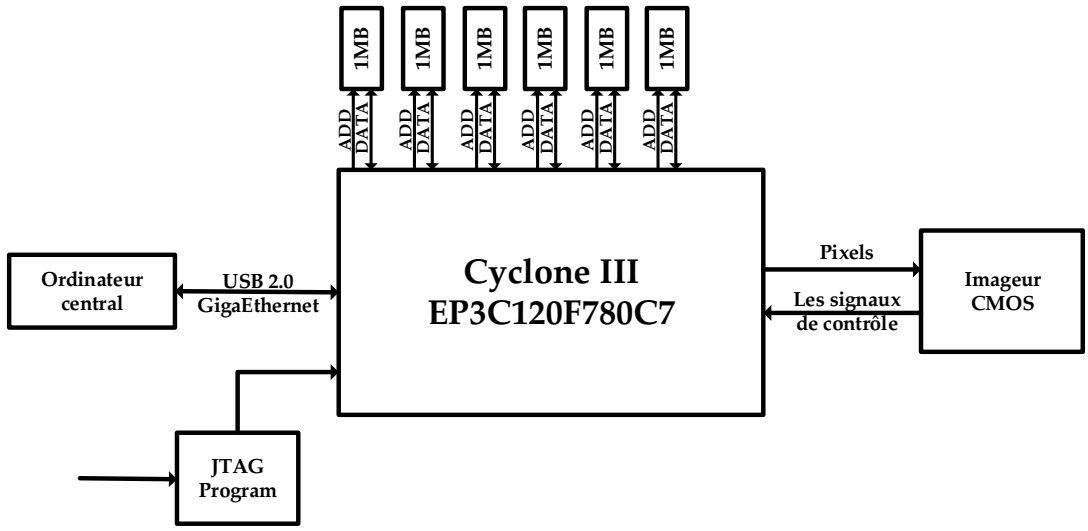


FIGURE 5.2: Synoptique de la *DreamCam*.

SRAM organisées en mots (1024K par 16bits) et ont une vitesse d'accès rapide de 8ns sous 3.3V et peuvent être facilement contrôlées.

CARTE D'ALIMENTATION : Les différentes cartes ont besoin de différents types de tension selon les dispositifs respectifs. La tension d'entrée initiale est de 6.5V et un ensemble de régulateurs génère les différentes tensions. La consommation globale est d'environ 1.4W lorsque le FPGA est non-programmé. Bien entendu, cette consommation varie largement avec la configuration du FPGA. Cette carte compte 18 tensions différentes de 1.2V à 5V ainsi que le programmeur JTAG[4] permettant la configuration du Cyclone-III.

Modèle	Altera Cyclone-III EP3C120
Logic Element (LE)s	119'088
M9K embedded memory blocks	432
Embedded memory (Kbits)	3'888
18-bit x 18-bit embedded multipliers	288
Phase Locked Loop (PLL)s	4
Maximum user I/O pins	531
Length × width (mm×mm)	23×23
Area (mm^2)	529

TABLE 5.1: Caractéristiques du dispositifs FPGA intégré dans la caméra intelligente DreamCam.

5.2.2 *Conception interne du FPGA*

Le but de la conception proposée est de créer une interface flexible entre la carte de détection et l'ordinateur. Dans cette approche, deux blocs sont très importants et doivent être utilisés dans chaque conception futur : le premier contrôle l'imageur CMOS, qui est le bloc « *Imag-IP* », le second gère la communication entre l'ordinateur et la *DreamCam*,

4 Joint Test Action Group (JTAG)

qui est le bloc « *Com-IP* ». Le bloc « *Mem-IP* » est utilisé lorsque des mémoires externes sont nécessaires. Ce bloc contrôle la mémoire en générant les signaux d'entrée approprié pour la mémoire (tels que le bus d'adresses A0 – A19, Chip Enable (CE), Write Enable (WE), Output Enable (OE)) et en réceptionnant les données qu'elle renvoie. Enfin, le bloc « *Algo-IP* » contient l'algorithme à implémenter sur le FPGA. Le schéma du système est représenté dans la Figure 5.13.

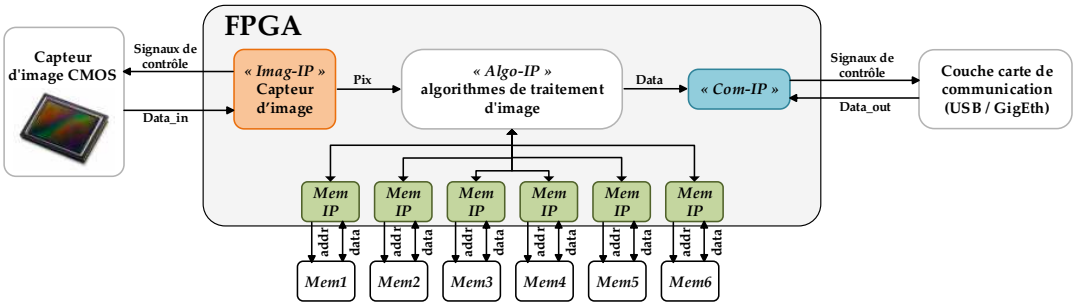


FIGURE 5.3: Conception interne du FPGA.

Ces différents blocs vont travailler comme suit : après la mise sous tension du système, l'imageur CMOS commence à fonctionner sous le contrôle de « *Imag-IP* » à l'intérieur du FPGA en envoyant les pixels de l'image un par un et ligne par ligne (débit noter « *Pix* » dans la Figure 5.13). Ces pixels sont envoyés au bloc « *Algo-IP* » de traitement d'images, où ils seront traités selon l'algorithme implémenté sur FPGA. Après cela, les résultats sont envoyés au bloc « *Com-IP* » (flux noter « *Data* » dans la Figure 5.13), où ils seront regroupés et envoyés vers l'ordinateur.

5.3 EXEMPLES D'ALGORITHMES IMPLÉMENTÉS SUR DREAMCAM

Pour tous les résultats présentés dans cette section, la *DreamCam* était équipée d'un imageur E2V et d'une carte de communication USB-2.0. La résolution de l'image était de 256×256 pixels. Le logiciel utilisé pour ces expériences était Quartus-II V13.0 avec les options de configuration par défaut.

5.3.1 *Détecteur de contour filtre de Sobel*

Le filtre de Sobel est couramment utilisé pour effectuer la détection de contours dans une image. Il constitue, en général, la première étape d'une chaîne de traitement. Ce filtre est implémenté comme une convolution de l'image avec deux masques de taille 3×3 (voir Équation 5.1).

$$M_x = \begin{Bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{Bmatrix} \qquad M_y = \begin{Bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{Bmatrix} \tag{5.1}$$

La convolution est une des opérations de bas-niveau. Selon le masque de convolution employé, différents résultats peuvent être obtenus (gradient vertical, horizontal, filtrage Gaussien, ...). L'Équation 5.2 représente la formule générale de l'opération, où A est une fenêtre de taille $(w \times w)$ autour du pixel (i,j) de l'image d'entrée, et B est le masque de convolution, également de taille $(w \times w)$:

$$Conv(i,j) = \sum_{n=i-\frac{w}{2},m=i-\frac{w}{2}}^{i+\frac{w}{2},i+\frac{w}{2}} \frac{1}{k(n,m)} A(n,m) \times B(n,m) \tag{5.2}$$

La matrice k est une matrice de pondération, dépendant de la nature et de la taille (w) du masque utilisé. Cette matrice peut être intégrée à la matrice B. L'opération décrite ci-dessus est appliquée pour chaque pixel de l'image d'entrée, et produit un pixel de l'image résultante.

Pour chaque pixel de l'image le résultat de l'opérateur de Sobel est la norme L1 du vecteur gradient lui correspondant. Cette opérateur a l'avantage de la simplicité de calcul, mais a une précision relativement faible, car il n'utilise que deux noyaux (suivant deux orientations 0° et 90°) de convolution pour détecter les contours dans une image. Pour le calcul de la norme du vecteur gradient d'un pixel de l'image $I(x,y)$, nous avons choisi pour l'implémentation hardware l'équation suivante :

$$I_{Sobel(x,y)} = |I(x,y) \otimes M_x| + |I(x,y) \otimes M_y| \tag{5.3}$$

Les contours de l'image sont obtenus en comparant les différentes normes à un seuil prédéfini. L'implémentation matérielle de cet opérateur de détection de contour revient à implémenter deux opérateurs de convolution avec les noyaux d'Équation 5.1.

Nous présentons ci-dessous le Tableau 5.2 de synthèse résumant les différentes ressources matérielles nécessaire pour l'implémentation de la convolution avec un filtre Gaussien. La fréquence maximum d'utilisation obtenue lors de la synthèse est de 218.39MHz. La Figure 5.4 donne un aperçu du résultat obtenu avec les filtre Sobel 3×3, pour la détection de contour.

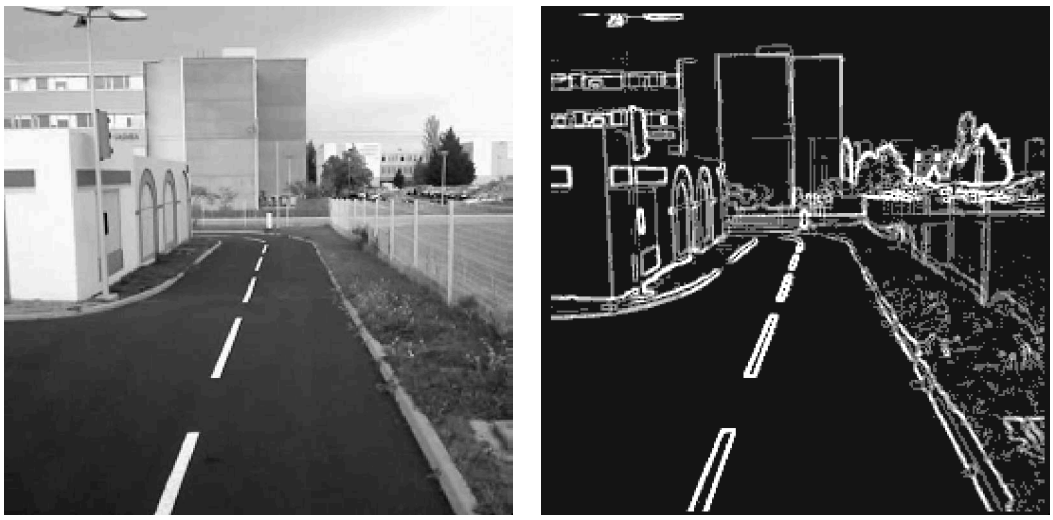
Type de Ressource	Occupation	
Total Logic Elements	293/119'088	(0.24%)
Total Memory Bits	4'032/3'981'312	(0.10%)
Embedded Multipliers	0/576	(0%)

TABLE 5.2: Le taux d'occupation sur le FPGA.

5.3.2 *Extracteur de primitives de type points d'intérêt*

Le mot *extracteur* de primitives est généralement utilisé pour décrire la combinaison d'un détecteur et d'un descripteur de primitives. Le détecteur est utilisé pour trouver les points d'intérêt (primitives) dans une image, tandis qu'un descripteur est utilisé pour décrire le voisinage entourant ces primitives.

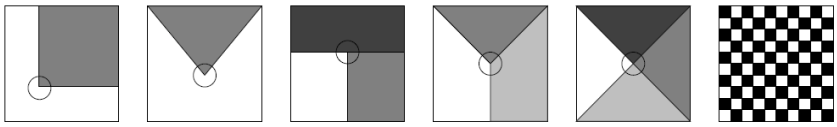
L'extracteur de caractéristiques présenté dans cette section est une combinaison d'un détecteur de primitive avec l'algorithme de Harris [68] et d'un descripteur simple qui donne pour chaque point d'intérêt un patch de l'image l'entourant.



(a) (b)

FIGURE 5.4: Exemple du résultat obtenu pour la détection de contour.

Les points d'intérêt sont définis comme étant des points qui possèdent des caractéristiques qui permettent de les distinguer des autres points de l'image, comme par exemple un fort contraste. Un point d'intérêt est associé à une discontinuité des niveaux de gris (voire des couleurs), de la texture, de la géométrie, ... Il est souvent assimilé à un coin. Les images de la Figure 5.5 donnent une panoplie de types de coins qui se situent à des jonctions de type « L », « V », « T », « Y », « X » et « damier ».



(a) (b) (c) (d) (e) (f)

FIGURE 5.5: Différents types de coins.

Les points d'intérêt ont la propriété d'être plus identifiables que les autres pixels de l'image, c'est pour cela qu'ils sont utilisés dans différentes applications : la robotique mobile, la reconstruction 3D, l'indexation ou la reconnaissance d'objets ou le suivi d'objets ... La Figure 5.6 montre un aperçu des résultats obtenus avec des primitives de type point d'intérêt. Les points (coins ou zone présentant un fort gradient) sont d'abord détectés, ensuite une petite fenêtre autour d'eux est gardée avec les coordonnées dans l'image. Cette opération est connue sous le nom *d'extraction de primitives*.

L'utilisation d'une telle opération permet de garder uniquement l'information la plus importante de l'image ce qui réduit la quantité de données à transmettre aux niveaux supérieurs. Plusieurs études comparatives entre les différentes primitives peuvent être trouvées dans la littérature, comme celle de Mikolajczyk and Schmid (2004) [126] et de Adam et al. (2012) [4].

De façon générale, un détecteur de point d'intérêt consiste à calculer en chaque pixel de l'image une grandeur appelée *valeur d'intérêt* qui dépend du pixel considéré et

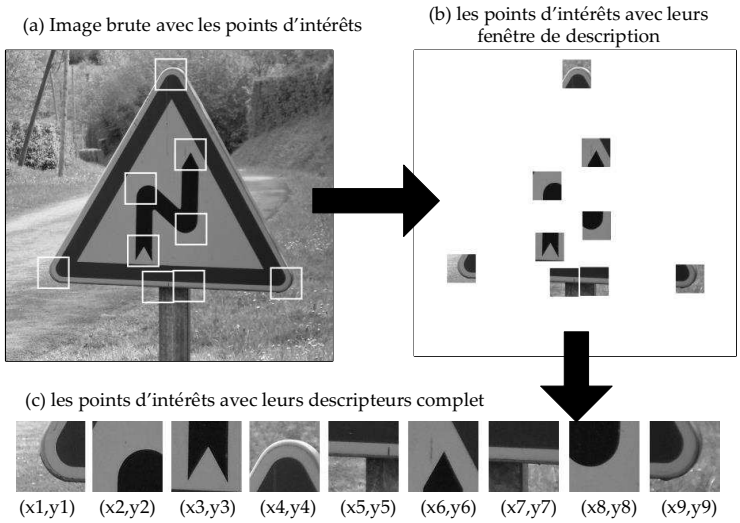


FIGURE 5.6: Principe de l'extraction de primitives : Dans cet exemple seul les points pertinents sont gardés. Une fenêtre d'intérêt autour de chaque point est découpée et le reste de l'image est ignoré.

de son voisinage. Cette valeur est ensuite seuillée pour savoir si le pixel est un point d'intérêt ou non.

Les algorithmes qui permettent d'extraire ces primitives sont souvent coûteux en temps de calcul, ce qui est un grand inconvénient lors du développement d'applications dites, temps réel. Parmi les solutions qui peuvent être envisagé pour remédier à ce type de problème est l'utilisation de matériel dédié à l'implémentation de ces algorithmes, comme par exemple, la *DreamCam*. Dans la section suivante nous présentons le détecteur que l'on a choisi pour la détection de points d'intérêt.

5.3.2.1 *Détecteur de Harris & Stephen*

Harris and Stephens (1988) [68] ont identifié certaines limitations dans le détecteur de Moravec (1980) [132] (voir Appendice A) et en les corrigeant, Harris and Stephens (1988) [68] ont déduit un détecteur de coins très populaire. Parmi ces limitations, il y a le caractère discret des directions de changement qui peuvent être effectués par des pas de 45°. Pour améliorer cet aspect, Harris and Stephens (1988) [68] ont considéré le développement de Taylor de la fonction d'intensité I au voisinage du pixel (u,v) :

$$I(x+u,y+v) = I(u,v) + x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta y} + o(x^2,y^2) \tag{5.4}$$

d'où :

$$E(x,y) = \sum_{u,v} w(u,v)[x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta y} + o(x^2,y^2)]^2 \tag{5.5}$$

Le paramètre $o(x^2,y^2)$ est négligé (valide pour les petits déplacements), ce qui donne :

$$E(x,y) = Ax^2 + 2Cxy + By^2 \tag{5.6}$$

avec :

$$\begin{cases} A &= (\frac{\delta I}{\delta x})^2 \bigotimes w \\ B &= (\frac{\delta I}{\delta y})^2 \bigotimes w \\ C &= (\frac{\delta I}{\delta x}\frac{\delta I}{\delta y}) \bigotimes w \end{cases} \tag{5.7}$$

La deuxième limitation concerne la fonction binaire $w(x,y)$ qui produit une réponse bruitée. Pour améliorer la réponse du détecteur, Harris and Stephens (1988) [68] ont proposé d'utiliser un filtre Gaussien :

$$w(x,y) = exp(\frac{-(x^2+y^2)}{2.\sigma^2}) \tag{5.8}$$

La troisième limitation prise en considération par Harris and Stephens (1988) [68] est le fait que la valeur d'intérêt d'un pixel n'est que le minimum de la fonction $E(i,j)$, ce qui représente une étude locale seulement. Harris and Stephens (1988) [68] proposent donc d'étudier le comportement général de la fonction $E(i,j)$ suivant les différentes directions, cela par l'étude de la matrice M :

$$E(x,y) = (x,y).\mathbf{M}.(x,y)^t \tag{5.9}$$

avec :

$$M = \begin{pmatrix} A & C \\ C & B \end{pmatrix} \tag{5.10}$$

La matrice M caractérise le comportement général de la fonction $E(i,j)$, les courbures principales de $E(i,j)$ correspondent aux valeurs propres λ_1 et λ_2 de cette matrice :

- Si les deux valeurs propres sont de faibles valeurs, alors les courbures de $E(i,j)$ le sont aussi donc le pixel traité appartient à une région uniforme,
- Si une valeur propre est plus grande que l'autre, alors il en sera de même pour la courbure, ce qui veut dire que le pixel traité est sur ou proche d'un contour,
- Si les deux valeurs propres sont grandes, les deux courbures sont de fortes valeurs, donc l'intensité varie fortement dans toutes les directions ce qui correspond à un point d'intérêt.

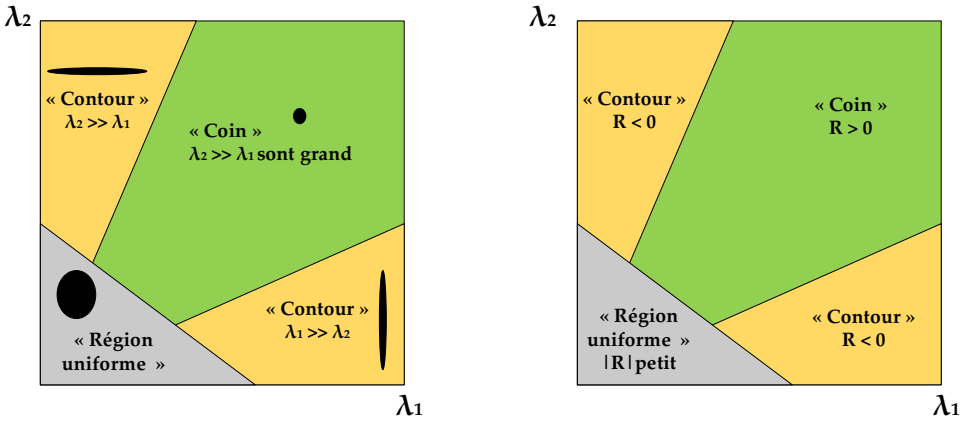
Pour éviter le calcul des valeurs propres, ***Harris & Stephen*** ont proposé l'opérateur suivant pour détecter les points d'intérêt dans une image, ce qui permet de calculer la valeur d'intérêt d'un pixel :

$$R = Det(M) - k * Trace(M)^2 \tag{5.11}$$

avec :

$$\begin{cases} k \in [0.04, 0.06] \\ Det(M) = AB - C^2 \\ Trace(M) = A + B \end{cases} \tag{5.12}$$

Les valeurs de R sont positives au voisinage d'un point d'intérêt, négatives au voisinage d'un contour et de faibles valeurs proches de zéro pour les zones homogènes d'intensité constante (voir Figure 5.7).



(a) Valeurs propres.

(b) Critère de Harris.

FIGURE 5.7: Détecteur de Harris & Stephen.

5.3.2.2 *Implémentation matérielle de l'extracteur de points d'intérêt*

Dans le détecteur de *Harris*, les principales opérations utilisées sont : la dérivée première et la convolution. Ces opérations sont des SIMD et donc très parallélisables, ce qui signifie qu'elles sont adaptées pour une implémentation sur FPGA.

Un pixel est considéré comme un point d'intérêt lorsque sa valeur d'intérêt R_i est supérieure au seuil prédéfini. Plus la valeur est élevée, plus le point d'intérêt détecté est précis. La valeur est calculée pour chaque pixel en utilisant la formule de l'Équation 5.11.

Le système d'extraction de caractéristiques proposé dans ce chapitre détecte d'abord les points d'intérêt sur une image, les tri et les décrit en utilisant un patch de pixels de l'image. Le système est composé de plusieurs modules (voir Figure 5.8) développés en VHSIC Hardware Description Language (VHDL) et qui sont entièrement compatibles avec une implémentation sur FPGA.

Les principaux modules du système sont les suivants :

LE MODULE « DÉTECTEUR » : Détecte les points d'intérêt en se basant sur l'algorithme de *Harris & Stephen* (la première partie de l'extracteur). La Figure 5.9 donne un aperçu de l'architecture utilisée pour l'implémentation de l'algorithme de *Harris* sur FPGA. Pour atteindre une fréquence plus élevée, le système doit être parallélisé au degré maximum permis par l'architecture de la caméra intelligente. Tel que représenter sur la Figure 5.9, toutes les opérations qui sont indépendantes les unes des autres ont été implémentées séparément. La performance du système peut également être augmentée en utilisant les blocs DSP pour toutes les multiplications et les additions qui se trouvent dans l'algorithme de détecteur de coin.

Le système reçoit le flux de pixels et place ces derniers un-par-un dans une pile First In First Out (FIFO). Le calcul de la valeur d'intérêt R_i démarre lorsque la pile FIFO est presque pleine plus précisément lorsque le deuxième pixel de la cinquième ligne est reçu. Après le calcul de la valeur d'intérêt R_i, un comparateur simple est utilisé pour déterminer si le pixel traité est un point d'intérêt ou non.

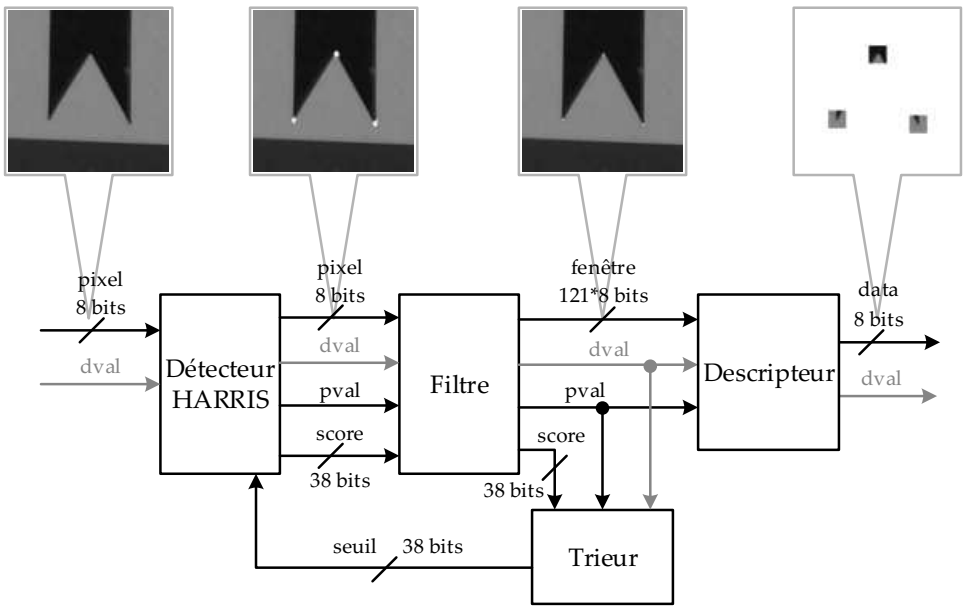


FIGURE 5.8: Architecture en quatre blocs de l'implémentation proposée de l'algorithme de Harris & Stephen. Une illustration de l'état de l'image à chaque étape du traitement est précisée sur la partie supérieure du schéma.

Dans ce module, les signaux les plus importants sont :

- « *pixel_i* » et « *dval_i* » en entrée, le premier représente le bus de données (codé sur 8bits) tandis que le second indique s'il y a des informations sur le bus de données.
- « *detect_o* » et « *score_o* » en sortie, le premier signal passe à « 1 » lorsque la valeur d'intérêt du pixel traité est supérieur au seuil prédéfini, ce qui veut dire que le pixel est un point d'intérêt, le second représente la valeur d'intérêt R_i du pixel correspondant codé sur 21bits. Cette valeur est gardée parce qu'elle sera utilisée dans les étapes suivantes.

LE MODULE « FILTRE » : Dans le premier module, les résultats de la détection sont filtrés, car quand une image est traitée en utilisant le détecteur Harris, plusieurs pixels autour d'un coin sont considérés comme des points d'intérêt. Le résultat souhaité est un seul point d'intérêt par coin, c'est-à-dire, un pixel pour chaque coin. La Figure 5.10 montre un exemple des résultats obtenus avec et sans ce module.

Dans le module implémenté dans nos travaux sur FPGA le filtrage est réalisé en conservant le premier pixel qui apparaît. Ce module est basé sur deux notions. Quand un point d'intérêt est détecté, le système vérifie s'il n'y a pas de point d'intérêt à proximité dans les « m » lignes précédentes. Si tel est le cas, le pixel sera défini comme un point d'intérêt et les « n » pixels suivants ne seront pas traités. S'il existe un point d'intérêt dans l'une des « m » lignes précédentes, le système passe au pixel suivant et ainsi de suite.

LE MODULE « TRI » : Les points d'intérêt sont triés dans l'ordre décroissant en fonction de leur valeur d'intérêt R_i. La technique employée pour trier les points d'inté-

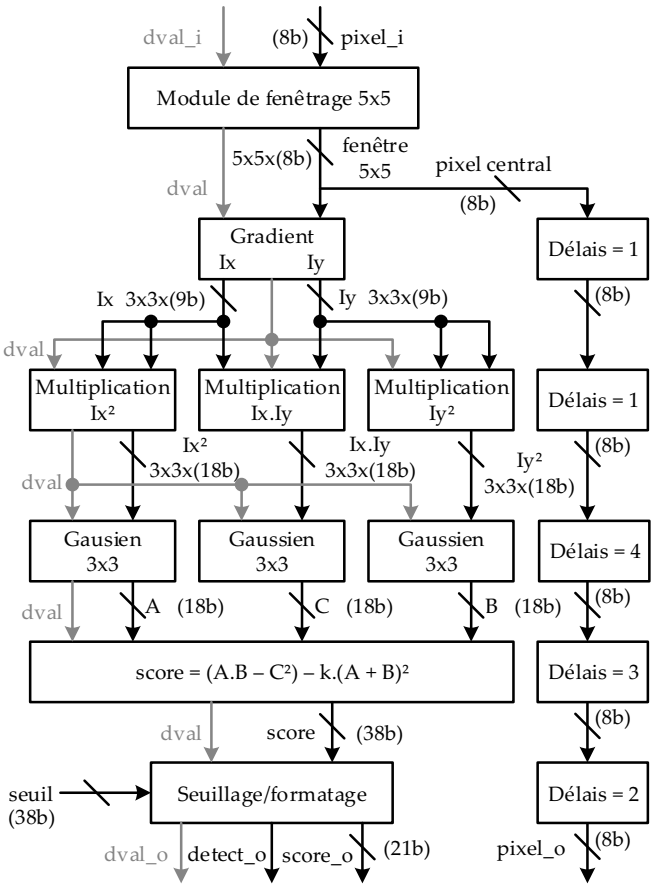


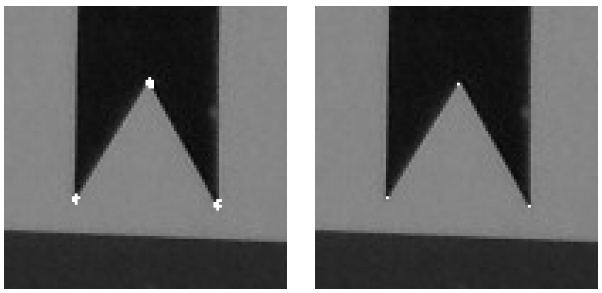
FIGURE 5.9: Architecture détaillée du module Détecteur de points d'intérêt.

rêt est celle décrite dans l'article de Yasuura et al. (1982) [199], elle se fait en deux étapes :

- Étape 1 : Dans cette étape l'ordre des points d'intérêt détectés est déterminé,
- Étape 2 : Les points sont placés dans une mémoire en fonction de l'ordre établit dans la première étape.

LE MODULE « DESCRIPTEUR » : Ce module représente la deuxième partie de l'extracteur. Ce module récupère le patch de chaque point d'intérêt et construit la trame de données contenant les coordonnées des points d'intérêt (X_i, Y_i) et leurs descripteurs (voir la Figure 5.11).

Ce module est composé de deux processus. Le premier construit la table qui contient les adresses mémoire des points d'intérêt détectés. Le second processus construit la trame de données en utilisant la mémoire mise en mode lecture et la table construite par le premier processus. Ce module contrôle deux mémoires asynchrones en contrôlant leurs différents signaux et leurs bus de données et d'adresses.



(a) La détection sans filtrage. (b) La détection avec filtrage.

FIGURE 5.10: Un exemple des résultats obtenus avec le module "Filtre" (des images en temps réel)

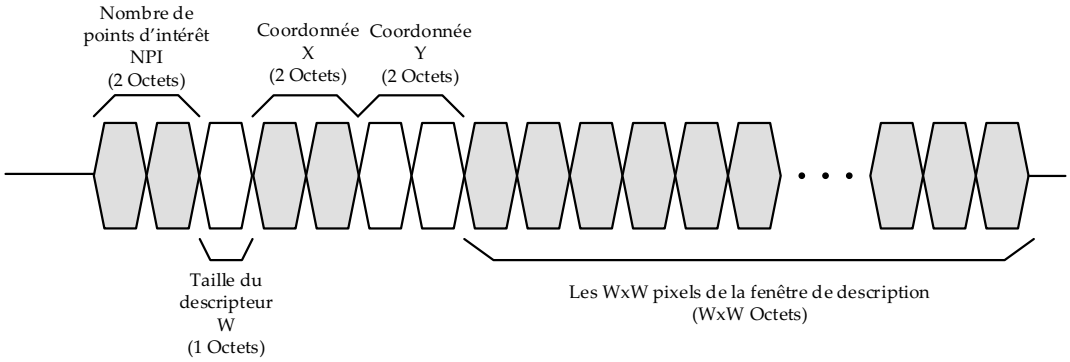


FIGURE 5.11: La trame de données construite par le bloc descripteur.

5.3.2.3 *Résultats d'implémentation*

Pour les résultats présentés dans cette section, la résolution de l'image était de 800 × 1024 pixels et la taille de chaque patch de point d'intérêt est de 15 × 15 pixels.

- *La consommation de ressources FPGA :*

  Un premier résultat concerne la consommation de ressources dans le FPGA. Le nombre de points d'intérêt que l'on veut obtenir influe directement sur la consommation des éléments logiques dans le FPGA. Cela est dû au module de description (nommé "Descripteur"), dont le rôle est de préparer la trame de données. C'est ce module qui utilise le plus de ressources internes.

  La Figure 5.12 montre que la consommation d'éléments logiques augmente de façon linéaire avec l'augmentation du nombre de points d'intérêt souhaités. Á titre d'information, le Tableau 5.3 donne toutes les ressources utilisées pour 200, 400 et 600 points d'intérêt.

- *Fréquence maximale :*

  Une deuxième mesure de performance de notre système réside dans la fréquence maximale permise. Le premier résultat est le fait que cette fréquence maximale dépend du nombre maximum de point d'intérêt détectable par le système. Le

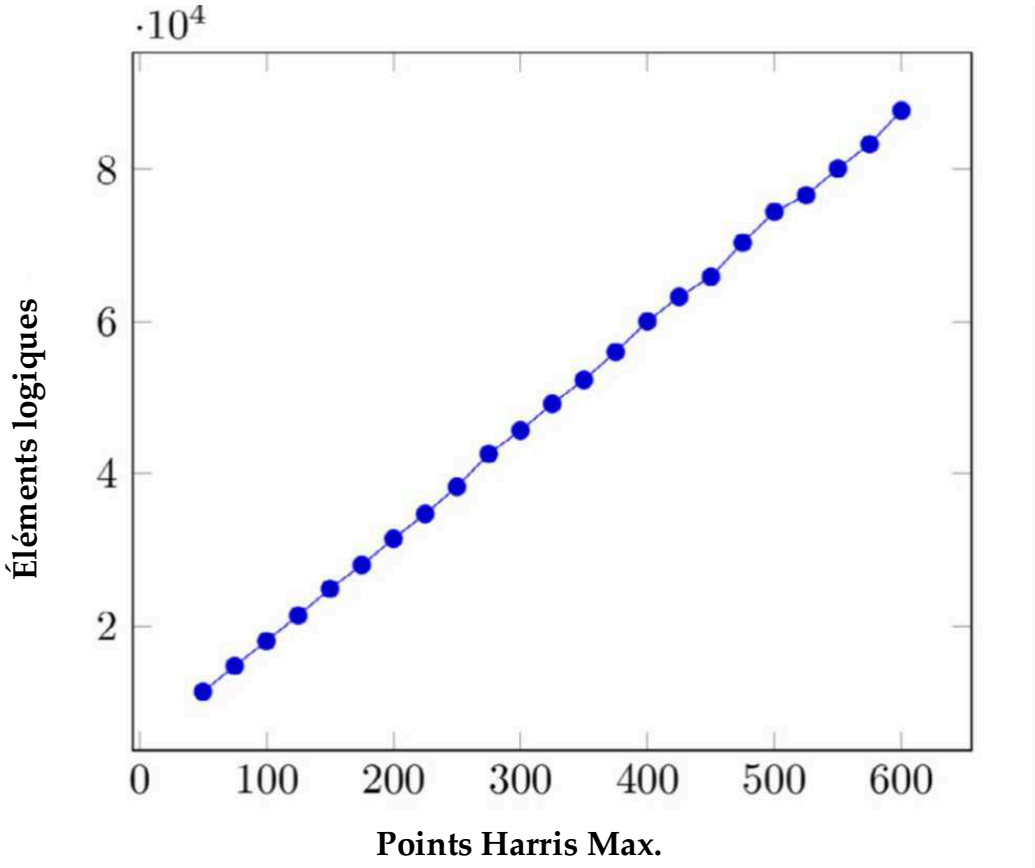


FIGURE 5.12: Éléments logiques utilisées en fonction du nombre de points d'intérêt souhaités. Pour cette expérience, un ensemble de synthèse a été réalisé à partir de 50 à 600 points par pas de 25 points. La résolution de l'image est de 800 × 1024.

second résultat est que la fréquence maximale décroît de 105MHz à 80MHz (Figure 5.13). Cette diminution est expliquée par l'augmentation de la longueur du chemin critique des données dans le module swap mémoire.

- *Expériences :*

  La Figure 5.14 présente les résultats obtenus lors du traitement des images capturées à partir d'un robot mobile dans des conditions expérimentales. Les images à gauche sont obtenues sans le module de description et les images à droite sont obtenues avec le module de description. La taille des images utilisées pour cette manipulation était de 256 × 256 pixels.

NOMBRE DE POINTS D'INTÉRÊT	ÉLÉMENT LOGIQUE 119'088	COMBINATOIRE 119'088	REGISTRES 119'088
200	31'433 (26.40%)	16'601 (13.94%)	23'014 (19.32%)
400	60'034 (50.42%)	29'121 (24.45S%)	43'814 (36.79%)
600	87'754 (73.69%)	41'636 (34.96%)	64'615 (54.26%)

TABLE 5.3: Ressources FPGA utilisées.

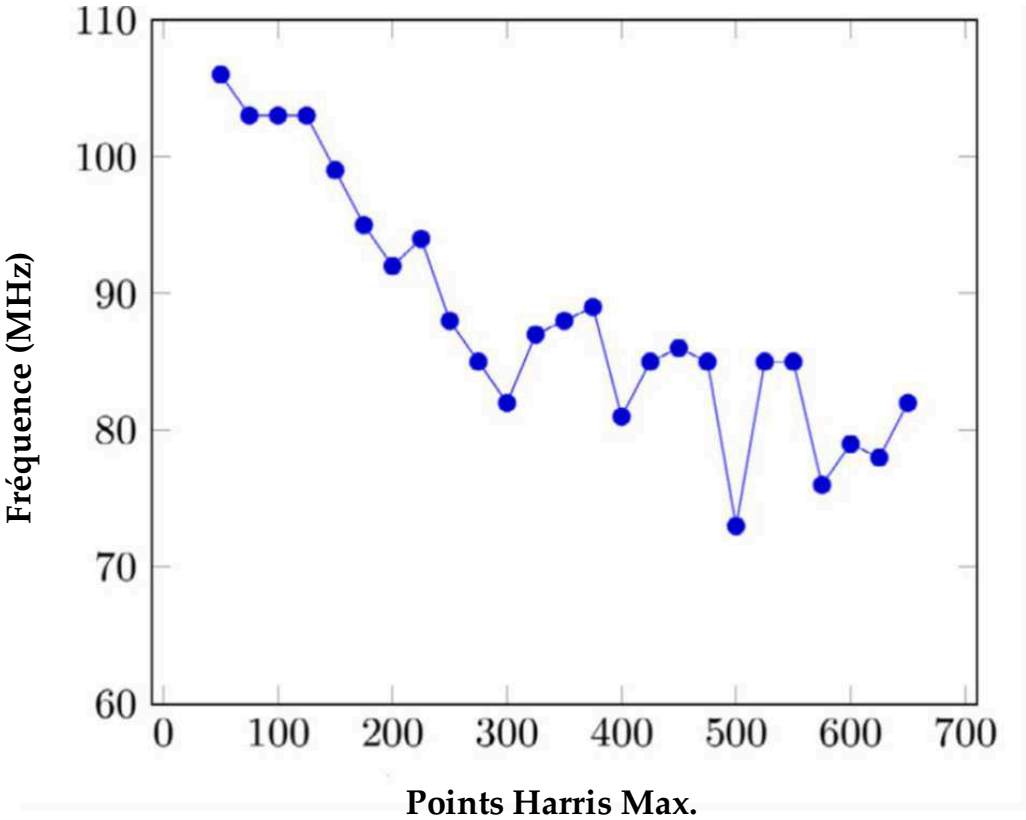


FIGURE 5.13: Fréquence maximale du système en fonction du NPI. Pour cette expérience, un ensemble de synthèses a été réalisé à partir de 50 à 600 points par pas de 25 points. La résolution de l'image est de 800 × 1024.

L'extracteur de primitives implémenté sur FPGA fonctionne sur le flux de pixels en provenance de l'imageur CMOS. La taille de la trame de données doit être inférieure ou égale à celle de l'image traitée, ce qui permettra à la trame de données d'être envoyée entièrement sans perte d'information.

En général, si les images traitées ont une taille de $L \times C$ pixels chacune et que le patch choisi pour décrire les points d'intérêt a une taille $W \times W$ pixels, le nombre maximum de points d'intérêt que le système peut détecter est de :

$$n < \frac{L \times C - 3}{W \times W + 4} \tag{5.13}$$

Où :

- Le -3 est pour les deux premiers éléments de la trame de données (le nombre de points d'intérêt détectés (2 octets) et la taille du patch (1 octet)),
- Le $+4$ est pour les deux coordonnées « X_o, Y_o » de chaque point d'intérêt, qui sont codées sur deux octets chacune,

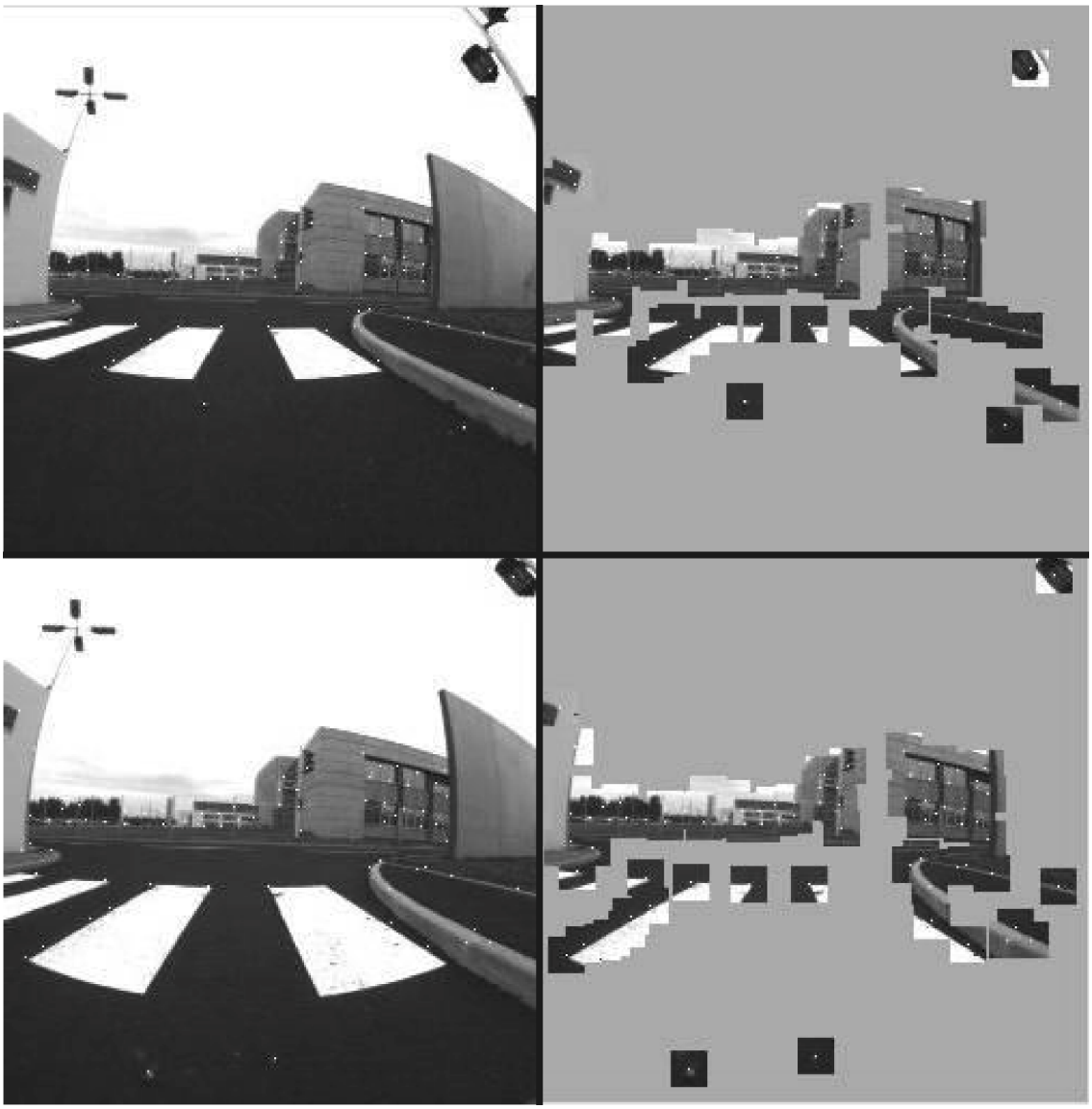


FIGURE 5.14: Les résultats obtenus lorsque le traitement est réalisé sur des images capturées à partir d'un robot mobile. À gauche la totalité de l'image avec les points d'intérêt en blanc. À droite les points d'intérêt et leur descripteur.

5.4 CONCLUSION

Le capteur embarqué présenté dans ce chapitre « *DreamCam* » peut être considéré comme une architecture réactive et surtout, en tant que plate forme de recherche pour capteur intelligent. Afin de montrer l'utilité et les performances d'une telle plate forme, nous avons présenté dans ce chapitre l'implémentation sur la *DreamCam* trois algorithmes de traitement d'images utilisé en robotique (L'algorithme de convolution par un filtre Gaussien, l'algorithme de détection de contour et l'algorithme d'extraction de primitives de type Harris & Stephen).

Dans le chapitre 6 c'est l'implémentation de l'algorithme de détection de régions saillante sur la *DreamCam* qui est présentée.

6

ARCHITECTURE POUR TRAITEMENT DE SAILLANCE

6.1 INTRODUCTION

Quand une personne voit une scène sans tâche spécifique à réaliser, elle a tendance à se concentrer sur les parties qui se démarquent le plus dans cette scène, c'est-à-dire, les régions saillantes. Le processus de détection de ces régions saillantes est principalement utilisé comme un algorithme de pré-traitement, réduisant la complexité ainsi que le nombre de calculs à effectuer dans les tâches suivantes de traitement. Par exemple, comme l'a montré Kim et al. (2009) [95] dans ses travaux, la détection de primitives sur certaines régions de l'image est plus rapide et plus efficace qu'une détection de primitives sur une image entière. Le principal inconvénient avec des modèles proposés pour l'extraction de zones de saillance est leur complexité de calcul, ce qui a conduit à des temps de traitement important.

Nous présentons dans ce chapitre l'architecture matérielle implémentée sur la caméra intelligente *DreamCam* de l'algorithme de détection de régions saillantes. Cet algorithme est utilisé pour la détection des régions saillantes qui se démarquent par rapport à leur voisinage. La détection de ces régions est une étape de pré-traitement qui permettra de concentrer les différents traitements qui suivent sur ces régions, ce qui réduit d'une façon considérable la quantité de données à transmettre (communication), la consommation d'énergie et aussi le temps de traitement.

Ce chapitre est organisé comme suit : dans la section 6.2, nous décrivons les implémentations déjà existantes d'algorithmes de détection de régions saillantes et une comparaison avec celle qu'on propose. La section 6.3 détaille l'architecture proposée pour l'implémentation du détecteur de région saillante choisi. Tandis que la section 6.4 évalue les performances de notre architecture. Enfin, une conclusion sur ce chapitre est donnée dans la dernière section.

6.2 TRAVAUX ANTÉRIEURS

Dans la littérature, quelques travaux comme ceux de Laurent et al. (2014) [104], Park et al. (2013) [154], Wang (2012) [190], Kestur et al. (2011) [93] et Bae et al. (2011) [10] ont déjà abordés la problématique de l'implémentation d'un détecteur de régions saillantes sur des plates formes à base de FPGA ou d'autres unités de traitement.

Laurent et al. (2014) [104] propose dans leurs travaux un prototype d'architecture matérielle sur un composant reconfigurable qui pourra extraire les régions saillantes à la même fréquence que la caméra utilisée. Dans ce travail la caractéristique *intensité* est la seule à être prise en considération, l'orientation et la couleur ne sont pas traitées. De plus cette implémentation n'a considéré que deux niveaux d'échelle pour la pyramide gaussienne sans aucun résultat visuel n'est donné.

Le travail présenté par Park et al. (2013) [154] dans leur article se focalise plus sur le lien qu'il y a entre la consommation d'énergie d'un système et sa précision d'extraction et de classification d'objet. La saillance est utilisée dans ce travail comme un bloc de pré-traitement, pour réduire la quantité de données à transmettre au bloc d'extraction et de classification d'objet, ce qui réduira la consommation d'énergie. Le modèle de saillance [26] utilisé dans ce travail est basé sur la théorie de l'information de Shannon [31]. Ce modèle considère uniquement deux caractéristiques : l'intensité et l'orientation. Les images traitées sont stockées en mémoire.

Dans les travaux de Wang (2012) [190] le détecteur de régions saillantes est utilisé comme un cas particulier pour l'étude d'architectures hybrides utilisant des parties créées avec des outils de description matérielle bas et haut niveaux. Le modèle [77] utilisé dans ces travaux pour l'extraction de régions saillantes se base sur la transformée de Fourrier. Les images traitées dans ce modèle sont en niveaux de gris et sont toutes ramenées à une taille très réduite de 64×64 pixels seulement. De même que précédemment, dans ce modèle les images sont aussi lues à partir d'une mémoire sur la carte de développement. Le FPGA utilisé est un Xilinx Virtex-6 fonctionnant à une fréquence de 100MHz.

Kestur et al. (2011) [93] présentent leur architecture Streaming Hardware Accelerator with Run-time Configurability (SHARC), dans laquelle la configuration du FPGA est faite de façon dynamique, c'est-à-dire pendant l'exécution d'une application. Cette architecture n'est pas dédié pour la détection des régions saillantes, mais afin de montrer les avantages de cette architecture et les avantages de la reconfiguration dynamique les auteurs ont choisi d'implémenter le modèle de Itti et al. (1998) [84] pour la détection des régions saillantes. Dans ce modèle deux filtres sont utilisées, celui de Gauss et de Gabor. La configuration dynamique de l'architecture SHARC est utilisée pour changer les filtres appliqués à l'image traitée.

Ces implémentations présentées ont toutes été faites sur des architectures qui ne sont pas connectées à des imageurs et qui utilisent des images stockées dans des mémoires. Ce que nous proposons dans ce chapitre est une implémentation qui utilise directement le flot de pixels arrivant de l'imageur, ce qui permet d'avoir un détecteur de régions saillantes ayant une fréquence image égale à celle de l'imageur (dans notre cas 45fps en pleine résolution c'est-dire 1280×960 pixels et 60fps en vidéo Haute Définition (HD) 720p60).

6.3 IMPLÉMENTATION MATÉRIELLE DE L'ALGORITHME DE SAILLANCE

Les modèles de détection de région saillantes proposés par Itti et al. (1998) [84], Frintrop (2006) [56] et autres, traitent généralement l'image d'entrée à des échelles multiples obtenues avec une pyramide gaussiennes dyadiques. L'implémentation matérielle d'une telle approche nécessite l'utilisation de plusieurs banques mémoires pour pouvoir accéder aux différents niveau d'image en même temps. La Figure 6.1 donne un aperçu de l'architecture à prévoir si l'implémentation matérielle se basait sur une telle approche.

Pour éviter l'utilisation excessive de mémoire, une autre approche sera implémentée se basant uniquement sur des filtres Gaussiens. Parce que les convolutions successives des images réduites avec un seul filtre Gaussien peuvent être approximées à des convolutions multiples par un filtre Gaussien ayant des largeurs différentes.

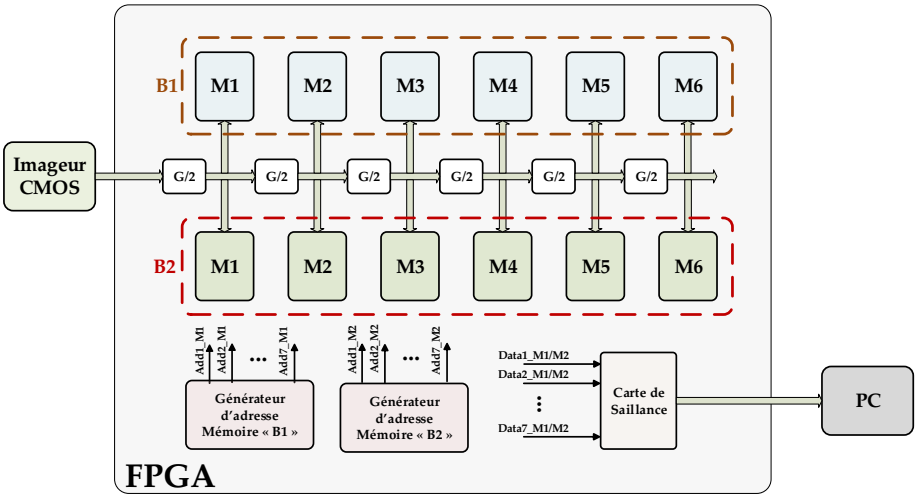


FIGURE 6.1: Architecture détection de saillance approche mémoire.

Les pixels des différents niveaux de la pyramide gaussienne seront approximativement équivalents aux réponses obtenues si des filtres Gaussien ayant des largeur de plus en plus grande étaient utilisés. La largeur du filtre Gaussien au niveau « $\sigma + 1$ » sera égale à $\sqrt{5}\times$ la largeur du filtre au niveau « σ » [78].

L'implémentation que nous proposons se base sur le modèle de Frintrop (2006) [56]. Cette implémentation se divise en plusieurs blocs comme le montre la Figure 6.2. La première unité de traitement de cette architecture est le « *Bloc Debayeur* », qui délivre à partir des pixels de l'imageur quatre signaux (I, R, G et B) représentant l'image en niveaux de gris (intensité) et les trois chaînes de couleur principales traitées dans le modèle choisi. Les cartes d'évidences : couleur et intensité sont calculées en utilisant des « *Bloc Gauss* ». Tandis que la carte d'évidence orientation est obtenue avec le « *Bloc Gabor* ». La carte de saillance finale « *S* » se calcule avec le « *Bloc Sommation Naïf* » qui calcule la moyenne des trois cartes d'évidence.

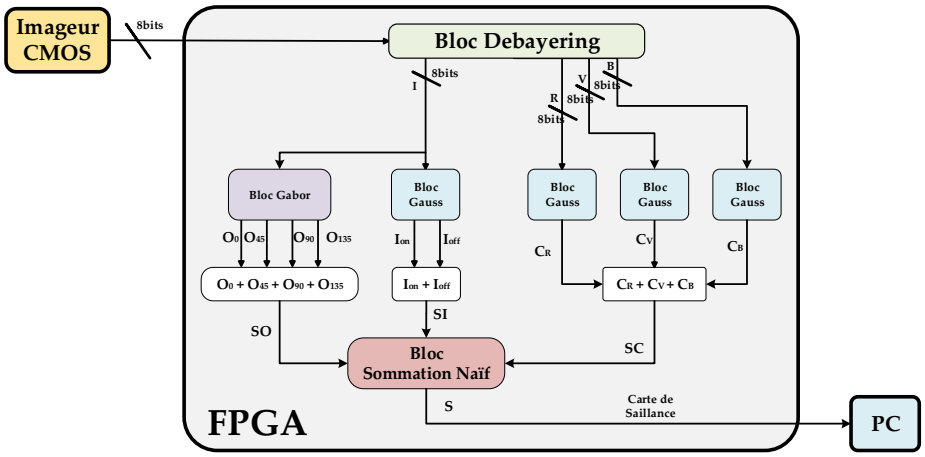


FIGURE 6.2: Architecture détection de saillance approche flot.

6.3.1 *Le Bloc Debayering*

La couleur représente un élément essentiel dans l'algorithme de détection de regions saillantes. Étant donné que la *DreamCam* utilise l'imageur MT9M031 qui contient un filtre de Bayer, un processus appelé dématriçage ou conversion des couleurs doit être fait pour retrouver les trois canaux de couleur RVB et l'image en niveaux de gris obtenu en calculant la moyenne des trois canaux RVB. Cette conversion est effectuée par interpolation des pixels adjacents.

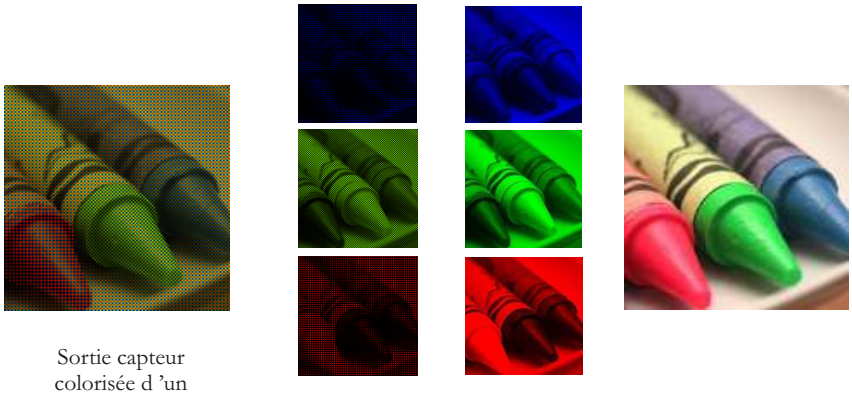


FIGURE 6.3: Exemple du résultat obtenu avec le processus de dématriçage.

Dans nos travaux nous avons choisi pour la suppression du filtre de Bayer l'algorithme d'interpolation bi-linéaire. Cet algorithme de dématriçage a besoin d'avoir accès à un bloc de pixels 3×3 pour effectuer les calculs nécessaires. Ce-ci nécessite de mettre en mémoire au minimum deux lignes et trois pixels de l'image traitée. Il existe d'autres algorithmes d'interpolation en traitement d'image, mais leur coût en terme de nombre de pixels requis et de complexité algorithmique est prohibitif pour une implémentation sur FPGA.

L'imageur MT9M031 donnant en sortie un pixel à la fois, codé sur 8bits, le flot de pixels remplit la pile FIFO utilisée comme mémoire tampon pour stocker les lignes nécessaires à l'algorithme de dématriçage. Les deux composantes couleurs manquantes d'un pixel $p(x,y)$ à calculer sont des fonctions des pixels environnantes $p(x_i,y_j)$, $i \times j = \{-1,0,1\} \times \{-1,0,1\}$. Au final les trois composantes (R,V,B) de chaque pixel traité auront un retard égal à la durée d'acquisition de 2 lignes et deux pixels par rapport au moment où le pixel correspondant a été capturé par l'imageur, cette technique de traitement ou d'implémentation de la fonction dématriçage permet d'avoir un système temps réel.

Dans le *Bloc Debayering* deux compteurs sont utilisés, un pour les lignes et un second pour les colonnes. La solution proposée utilise le bit de poids le plus faible des deux compteurs de lignes et de colonnes pour déterminer la couleur du pixel en cours de traitement et les fonctions à utiliser pour déterminer les deux couleurs manquantes. Le Tableau 6.1 résume les quatre cas possible.

Les pixels verts de type 1 et 2, nécessite des fonctions différentes pour le calcul des composantes rouge et bleu. Parce que les pixels vert de type 1 sont entre deux

Cmpt Lin bit-o	Cmpt Col bit-o	Couleur $Pix(0,0)$	Première couleur	Seconde couleur
0	0	Vert-1	Rouge = $\frac{1}{2}\sum_{i\in L1} Pix(i,0)$ $L1=\{-1,+1\}$	Bleu = $\frac{1}{2}\sum_{j\in L1} Pix(0,j)$ $L1=\{-1,+1\}$
0	1	Rouge	Vert = $\frac{1}{4}\sum_{(i,j)\in L2} Pix(i,j)$ $L2=\{(-1,0),(0,-1),(0,+1),(+1,0)\}$	Bleu = $\frac{1}{4}\sum_{(i,j)\in L2} Pix(i,j)$ $L2=\{(-1,-1),(-1,+1),(+1,-1),(+1,+1)\}$
1	0	Bleu	Vert = $\frac{1}{4}\sum_{(i,j)\in L2} Pix(i,j)$ $L2=\{(-1,0),(0,-1),(0,+1),(+1,0)\}$	Rouge = $\frac{1}{4}\sum_{(i,j)\in L2} Pix(i,j)$ $L2=\{(-1,-1),(-1,+1),(+1,-1),(+1,+1)\}$
1	1	Vert-2	Rouge = $\frac{1}{2}\sum_{j\in L1} Pix(0,j)$ $L1=\{-1,+1\}$	Bleu = $\frac{1}{2}\sum_{i\in L1} Pix(i,0)$ $L1=\{-1,+1\}$

TABLE 6.1: La couleur du pixel en cours de traitement.

pixels rouges (horizontalement), alors que ceux du type 2 sont entre deux pixels bleus (horizontalement).

La Figure 6.4 résume l'architecture matérielle implémentée sur la *DreamCam*, qui donne les trois composantes couleurs (R,V,B) de l'image ainsi que l'image en niveaux de gris.

6.3.2 *Le Bloc Gauss*

Dans ce bloc nous avons implémenté l'approche des filtres Gaussiens successifs avec différentes largeurs. La Figure 6.5 représente l'architecture de ce bloc, les différentes valeurs des filtres Gaussien $\sigma_i, i \in \{1\ldots7\}$ sont pré-calculées sur PC ensuite directement transmises aux différents blocs correspondants sous forme de constantes. Chaque sous bloc de cette architecture contenant un filtre Gaussien a besoin d'avoir accès à un bloc de pixels 3×3 pour effectuer l'opération de convolution. Ce qui nécessite l'utilisation d'une pile FIFO pour stocker deux lignes et deux pixels au minimum.

Comme le modèle choisi pour la détection des régions saillantes est celui de Frintrop (2006) [56], le bloc « OPER Frint » aura deux sorties à chaque niveau pour « *on-center* » et « *off-center* », ce qui donne à la fin pour chaque caractéristique (intensité et couleurs RVB) traitées par ce bloc deux cartes d'évidences (ON-OFF). Ces cartes sont additionnées par la suite deux à deux.

6.3.3 *Le Bloc Gabor*

Nous appelons filtres de Gabor l'association d'une courbe de Gauss et d'une sinusoïde orientée (voir Figure 6.6). En traitement d'images, nous travaillons dans le domaine

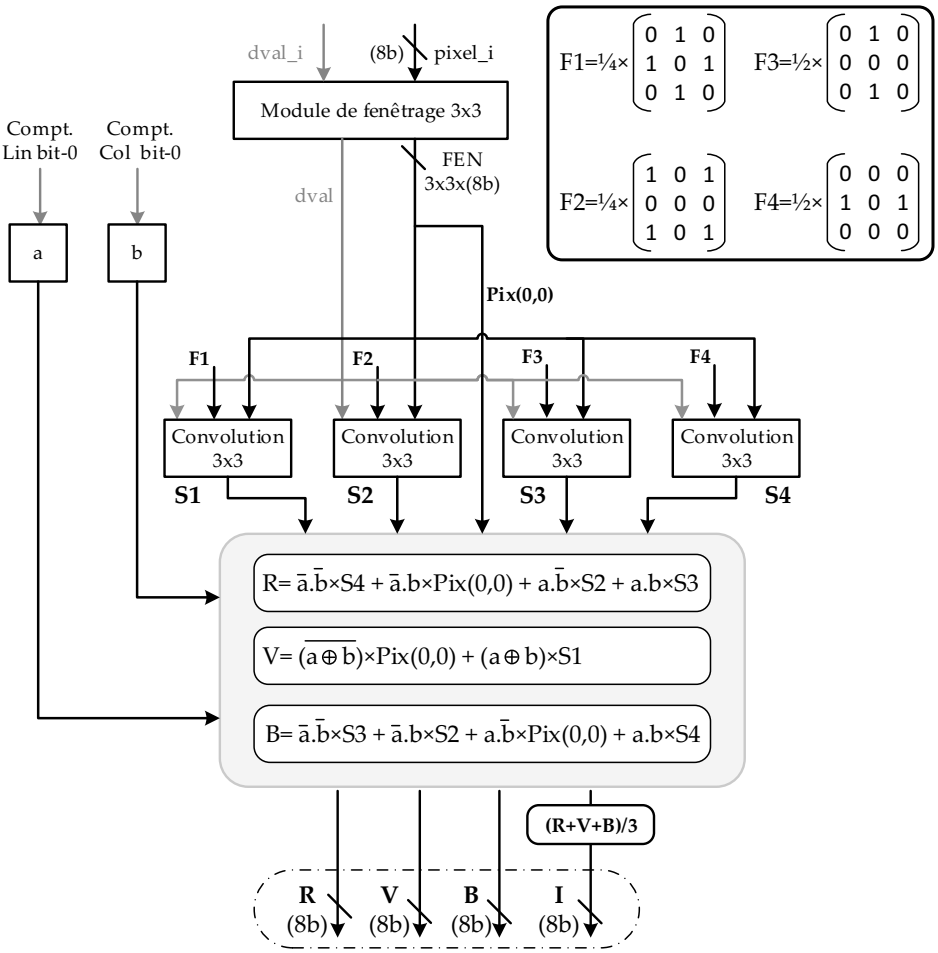


FIGURE 6.4: Architecture détaillée du Bloc Debayeur.

spatial en dimensions deux, ce qui permet d'écrire la fonction du filtre Gabor de la manière suivante :

$$G(x, y, \theta, f) = \cos(2\pi f x_\theta) \exp^{\left(-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right)\right)} \tag{6.1}$$

avec :

$$\begin{cases} x_\theta = x\cos\theta + y\sin\theta \\ y_\theta = y\cos\theta - x\sin\theta \end{cases} \tag{6.2}$$

où θ est l'orientation de la sinusoïde, f sa fréquence et σ_x (respectivement σ_y) l'écart-type de la gaussienne selon l'axe des abscisses (resp. des ordonnées). En appliquant cette fonction à un masque de convolution, on définit un filtre de convolution que nous appelons filtre de Gabor (voir Figure 6.7). Les filtres de Gabor permettent d'isoler les contours d'une image d'orientation perpendiculaire à θ et répondant à une certaine épaisseur, qui dépend de f.

L'implémentation de ce bloc sur FPGA est faite en se basant sur le même principe utilisé dans le bloc précèdent. Les valeurs des filtres Gabor pour $\sigma_i, i \in \{1 \ldots 6\}$ et $\theta \in$

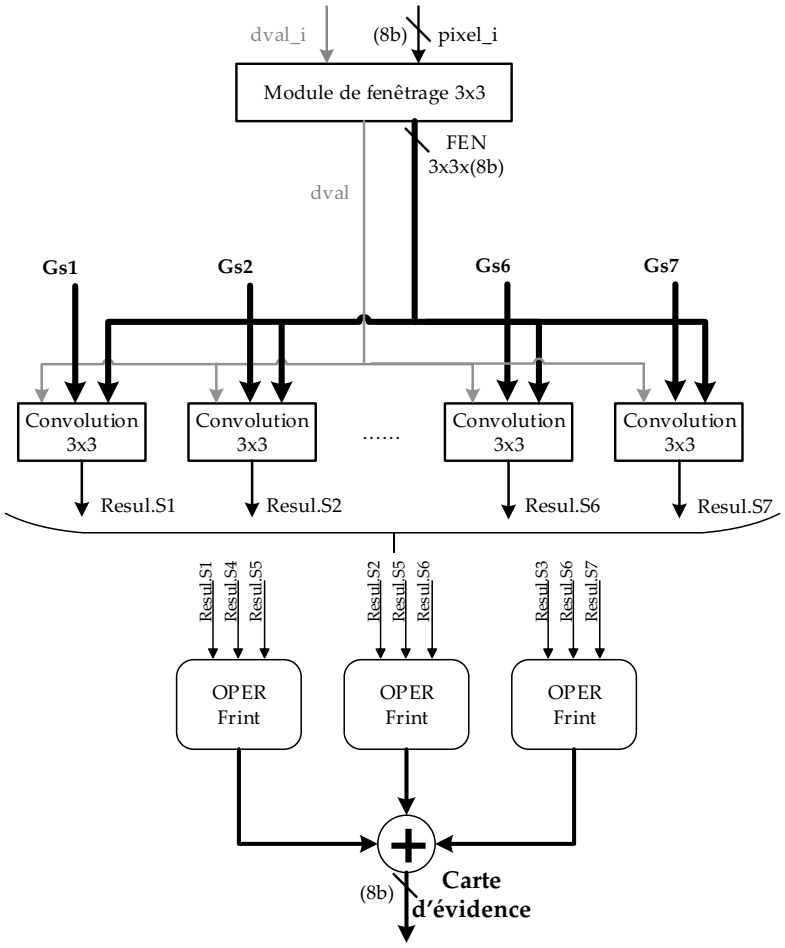


FIGURE 6.5: Architecture détaillé du Bloc Gauss.

$\{0°, 45°, 90°, 135°\}$ sont toutes pré-calculées sur PC puis transmises aux différents blocs comme étant des constantes. La taille des filtres de Gabor choisie est 7, correspondant à la taille minimum pour une implémentation sur FPGA suivant les quatre orientations. Ceci signifie que la pile FIFO utilisée dans ce bloc aura une taille minimale de sept lignes et six pixels.

Ce bloc donne au final la carte d'évidence correspondante à chaque orientation séparément. L'opération *centre-contour* est incluse directement dans les valeurs des filtres de Gabor pré-calculées. La Figure 6.8 résume le principe de fonctionnement du bloc. Finalement, La carte d'évidence de la caractéristique orientation O est obtenue par la sommation des quatre cartes d'évidence des orientations $\theta \in \{0°, 45°, 90°, 135°\}$.

6.4 LES RÉSULTATS EXPÉRIMENTAUX

Cette section présente les résultats d'implémentation du détecteur de régions saillantes sur FPGA. Pour tous les résultats présentés dans cette section, la *DreamCam* était équipée

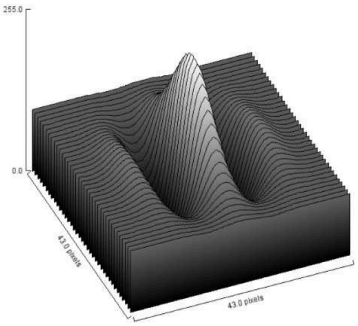


FIGURE 6.6: Représentation tridimensionnelle de la fonction de Gabor normalisée entre les valeurs 0 et 255.

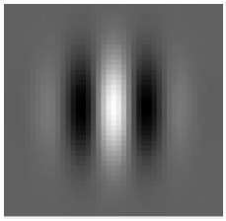


FIGURE 6.7: Masque du filtre de Gabor de rayon 21 pixels pour $\theta = 0$, $f = \sqrt{2}/10$ et $\sigma_x = \sigma_y = 7$.

d'un imageur couleur MT9 et d'une carte de communication USB-2.0. La résolution de l'image est de 512×1024 pixels et la taille des filtres Gaussian est de 3×3, tandis que des filtres de Gabor est de 7×7. Le logiciel utilisé pour ces expériences était Quartus-II V13.0 avec les options de configuration par défaut.

6.4.1 *La consommation de ressources FPGA*

Un premier résultat concerne la consommation des ressources dans le FPGA. L'occupation totale du détecteur de régions saillantes en considérant les trois primitives (intensité, couleur et orientation) dans le FPGA est présentée dans le Tableau 6.2.

Bloc	Type de Ressource	Occupation	
Bayer	Total Logic Elements	542/119'088	(1%)
	Total Memory Bits	16'320/3'981'312	(1%)
	Embedded Multipliers	0/576	(0%)
4×Gauss	Total Logic Elements	4×2'120/119'088	4×(2%)
	Total Memory Bits	0/3'981'312	(0%)
	Embedded Multipliers	0/576	(0%)
Gabor	Total Logic Elements	47'290/119'088	(40%)
	Total Memory Bits	48'720/3'981'312	(2%)
	Embedded Multipliers	0/576	(0%)

TABLE 6.2: Le taux d'occupation de chaque bloc sur le FPGA.

Le bloc qui consomme le plus de resources est celui implémentant le filtre de Gabor car dans cette implémentation des filtres de tailles 7×7 sont utilisés. Ce bloc contient les quatre orientations $(0°, 45°, 90°, 135°)$. Il faut savoir que dans l'implémentation finale un seul bloc Gabor a été utilisé, tandis que pour le bloc Gauss quatre duplications de ce bloc ont été faite, car le même traitement est fait pour les trois cartes couleur RVB et à l'image en niveaux de gris (intensité).

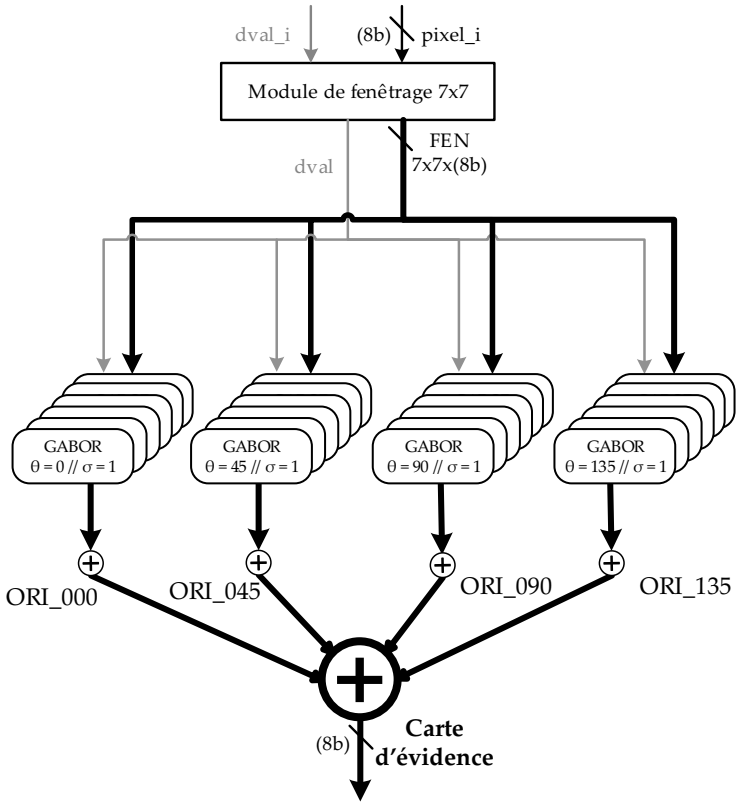


FIGURE 6.8: Architecture détaillé du Bloc Gabor.

6.4.2 *Fréquence maximale*

Une deuxième mesure de performance de notre système réside dans la fréquence maximale permise. Comme on désire avoir un système travaillant sur le flot pixels de l'imageur, les fréquences maximales des différents blocs constituant notre système doivent être supérieures à celle de l'imageur. D'après le fabricant de l'imageur MT9 cette fréquence peut être choisie entre $6-50$MHz. Dans nos expérimentations nous avons choisi la valeur de 33MHz. Le Tableau 6.3 regroupe les fréquences maximales des différents blocs utilisées (Bayer - Gauss - Gabor) séparément.

Bloc	Bayer	Gauss	Gabor	Saillance
Fréquence (MHz)	171.14	58.77	37.16	36.96

TABLE 6.3: La fréquence maximale de chaque bloc.

6.4.3 *Comparaison*

Le Tableau 6.4 donne une comparaison des performances de notre système avec les implémentations existantes (voir section 6.2). Ce tableau fournit également une comparaison de la consommation d'énergie en terme du nombre d'images traité par Watt.

Implémentations	Modèle	Matérielle	Fréquence	FPS	Puissance (W)	FPS/Watt
Impl. PC [157]	[84]	Processeur Intel Xeon	2.80 GHz	19	80	0.25
Impl. GPU [196]	[84]	4 Nvidia GeForce 8800(GTX)	1.35 GHz	313	155	0.61
Impl. FPGA [104]	[104]	Xilinx Zynq 7020	-	> 60	-	-
Impl. FPGA [154]	[26]	Xilinx Virtex-6 SX475T	-	-	372	-
Impl. FPGA [190]	[77]	Xilinx Virtex-6	100 MHz	-	-	-
Impl. FPGA [93]	[84]	Xilinx Virtex-6 LX240T	200 MHz	90	10	8.96
Notre impl. FPGA	[56]	Altera Cyclone-III	33 MHz	45	3 - 4	11.25 - 15

TABLE 6.4: Comparaison de notre système avec les implémentation précédentes.

Comme on peut le voir du tableau ci-dessus, grâce à notre système implémenté sur la caméra intelligente *DreamCam*, nous obtenons le meilleur résultat en nombre d'image traité par Watt.

6.4.4 *Expériences*

Dans ce qui suit nous donnons les images représentant les cartes de saillance et d'évidence de trois scènes différentes. La première est une image synthétique utilisé principalement pour les caractéristiques de couleurs et orientations. Les autres images représentent deux lieux différents, intérieur (une salle) et extérieur (du circuit PAVIN). Ces trois images sont représentées sur la Figure 6.9.

FIGURE 6.9: Les trois images utilisées dans les différents tests.

Les résultats sur les pages suivantes sont composées de trois colonnes. Les images de la colonne gauche représentent les différents résultats obtenus avec le modèle de détection de région saillantes implémentées sur la caméra intelligente *DreamCam*. Les images de la colonne du milieu représentent les résultats obtenus avec Matlab avec le même modèle. La dernière colonne (à droite) représente les résultats obtenus avec le modèle de Frintrop (2006) [56] complet implémenté sur Matlab.

6.5 CONCLUSION ET PERSPECTIVES

Nous avons présenté dans ce chapitre l'implémentation du détecteur de régions saillantes sur FPGA. Le système proposé sur la *DreamCam* détecte les régions saillantes à la fréquence de l'imageur. Pour l'instant seule l'étape de détection est implémentée, avec un module de sommation de cartes naïf sans aucune pondération.

La prochaine étape améliorant notre implémentation est l'ajout d'un module permettant de faire la normalisation de toutes les cartes en temps réel, et un autre pour la pondération des différentes cartes d'évidence ce qui améliora le résultat final.

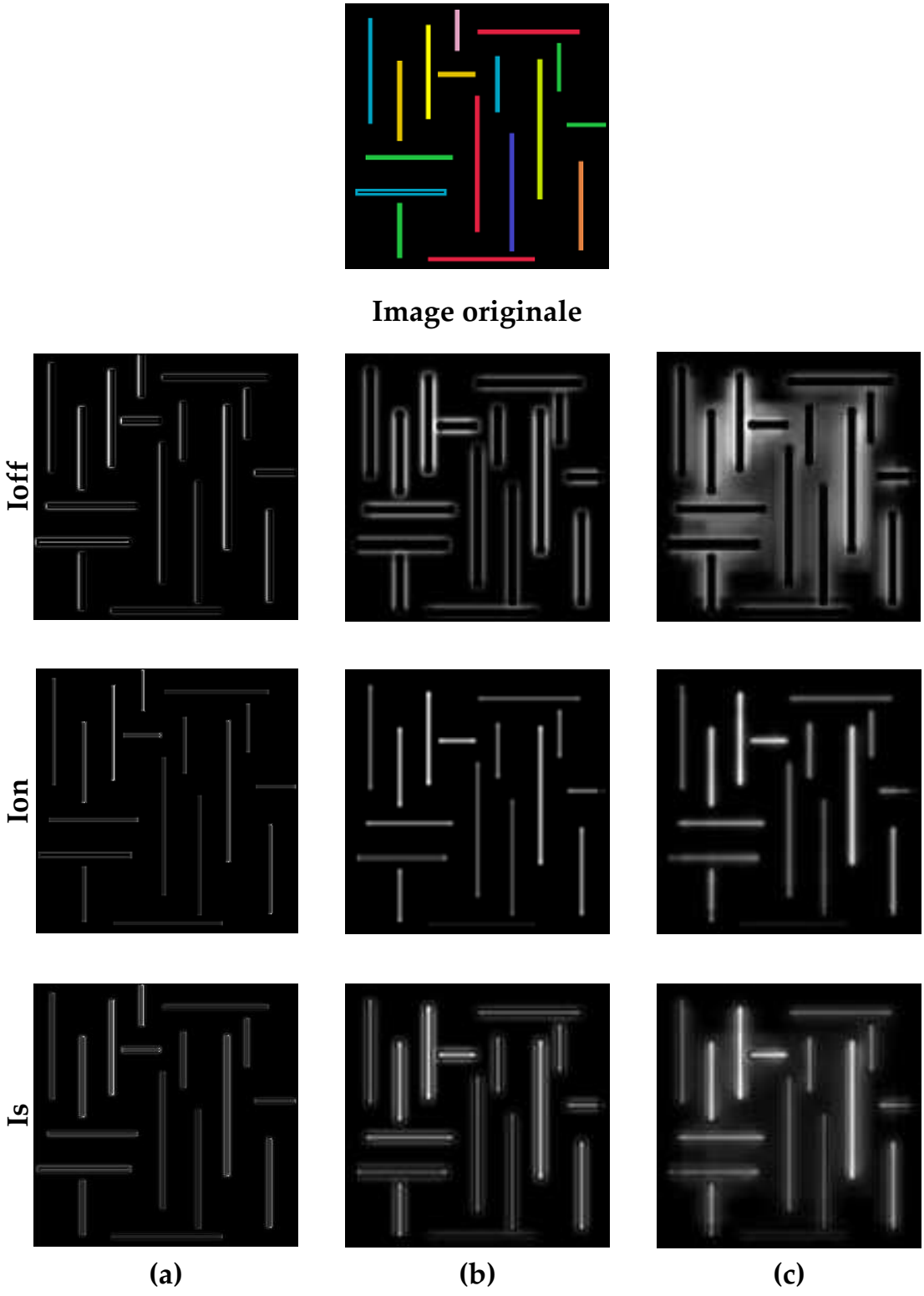


FIGURE 6.10: Les cartes d'évidences intensités (Image couleur).

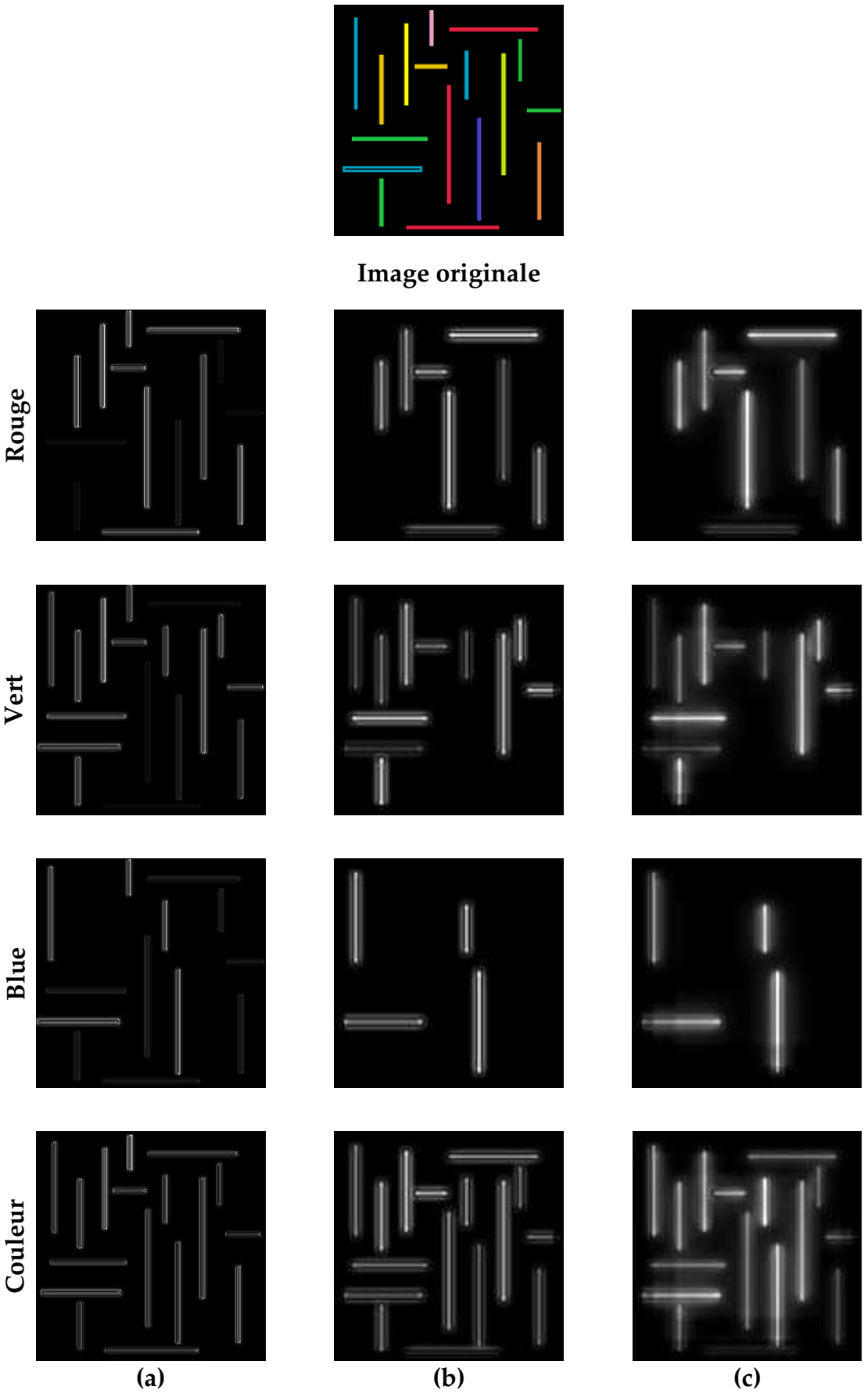


FIGURE 6.11: Les cartes d'évidences couleurs (Image couleur).

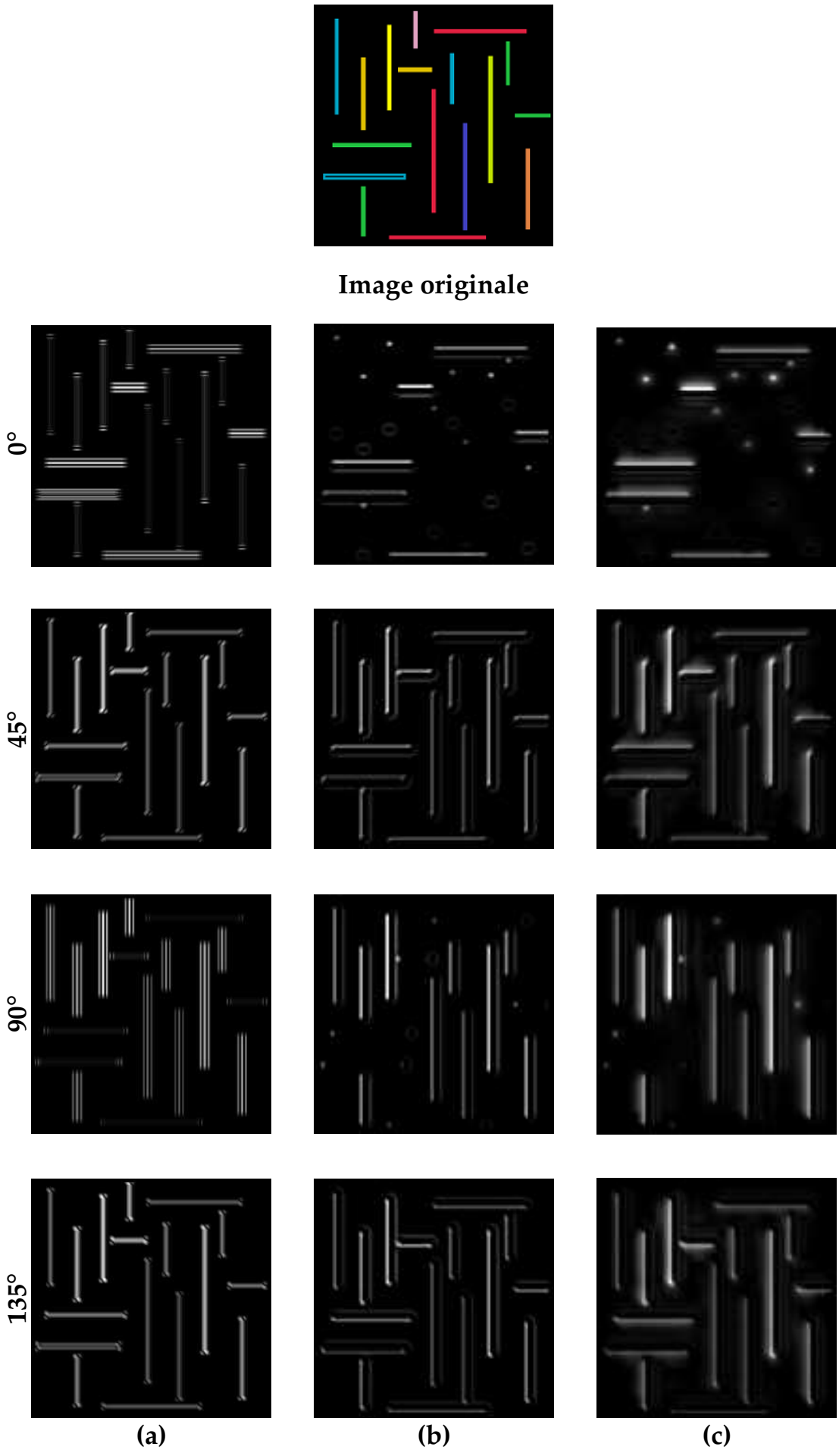


FIGURE 6.12: Les cartes d'évidences orientation (Image couleur).

FIGURE 6.13: Les cartes d'évidences intensités (Image bureau).

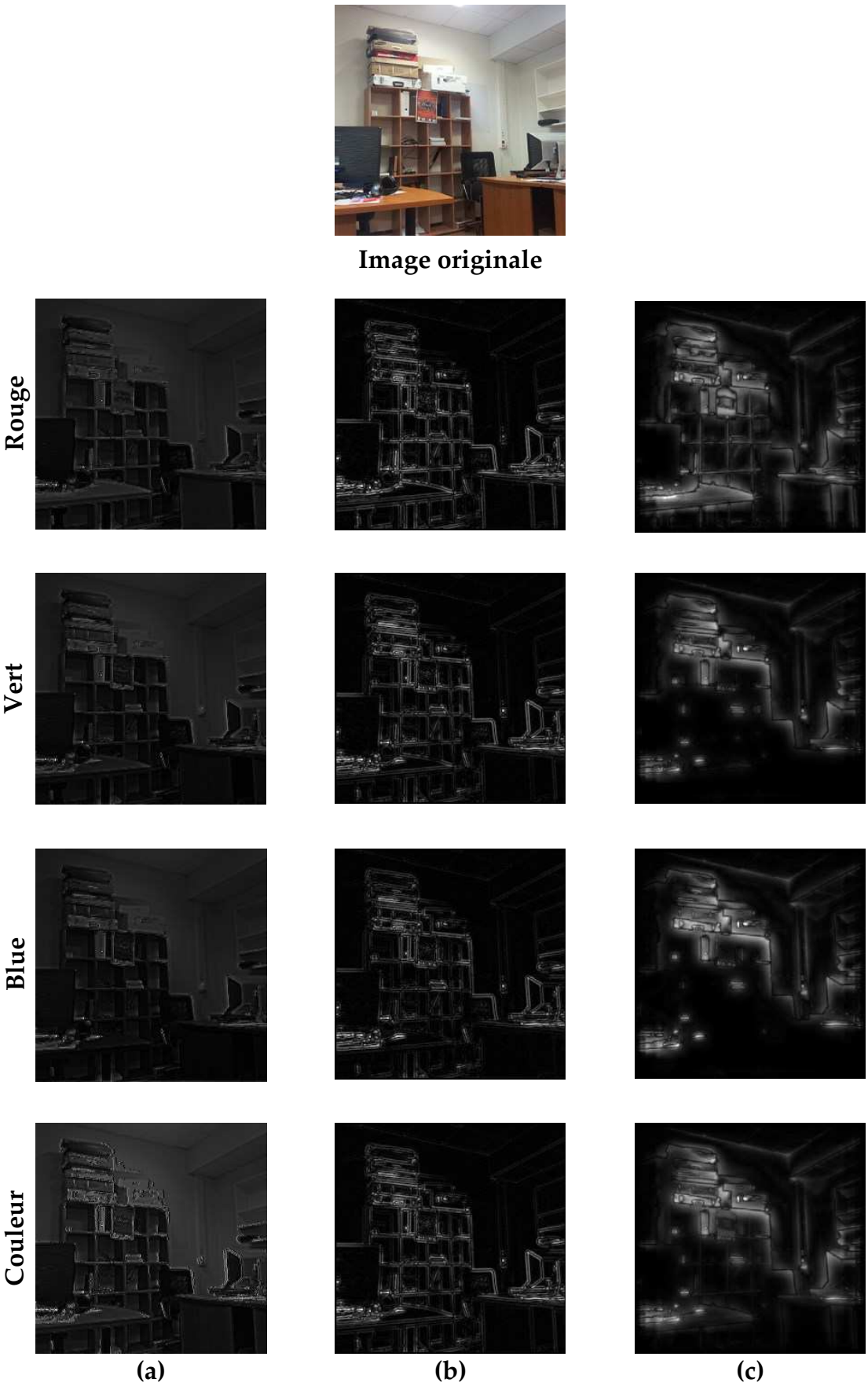


FIGURE 6.14: Les cartes d'évidences couleurs (Image bureau).

FIGURE 6.15: Les cartes d'évidences orientation (Image bureau).

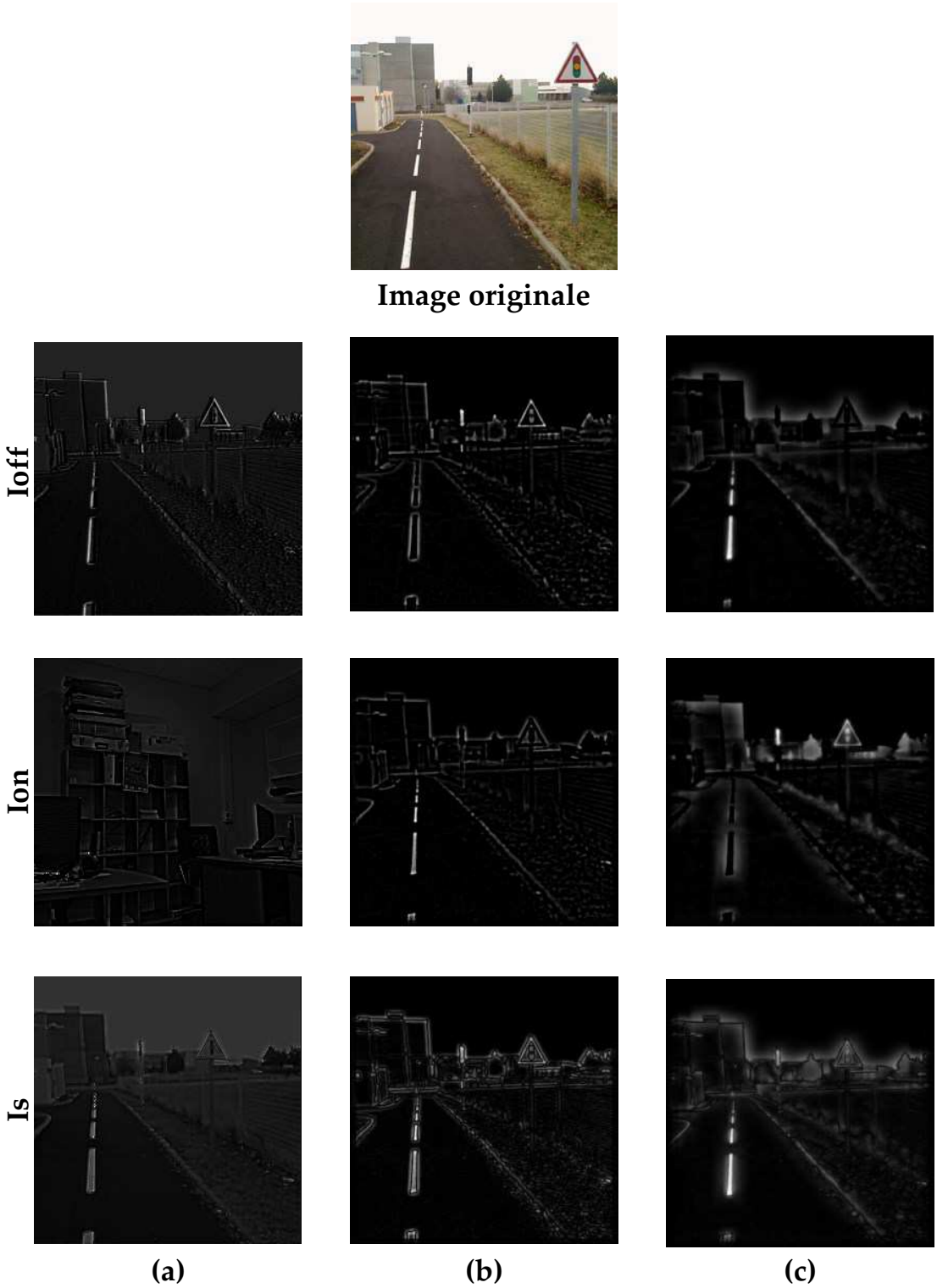


FIGURE 6.16: Les cartes d'évidences intensités (Image PAVIN).

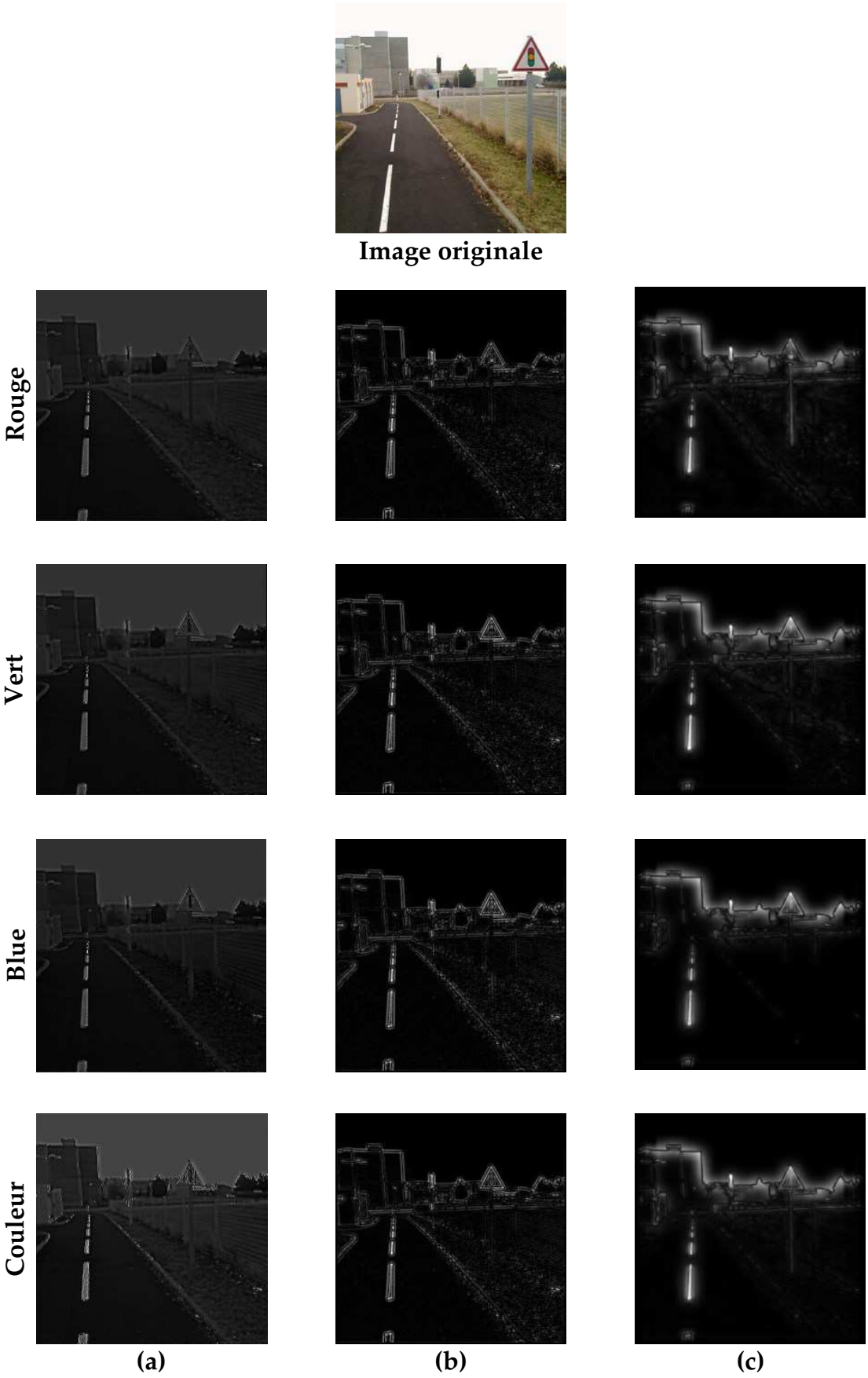


FIGURE 6.17: Les cartes d'évidences couleurs (Image PAVIN).

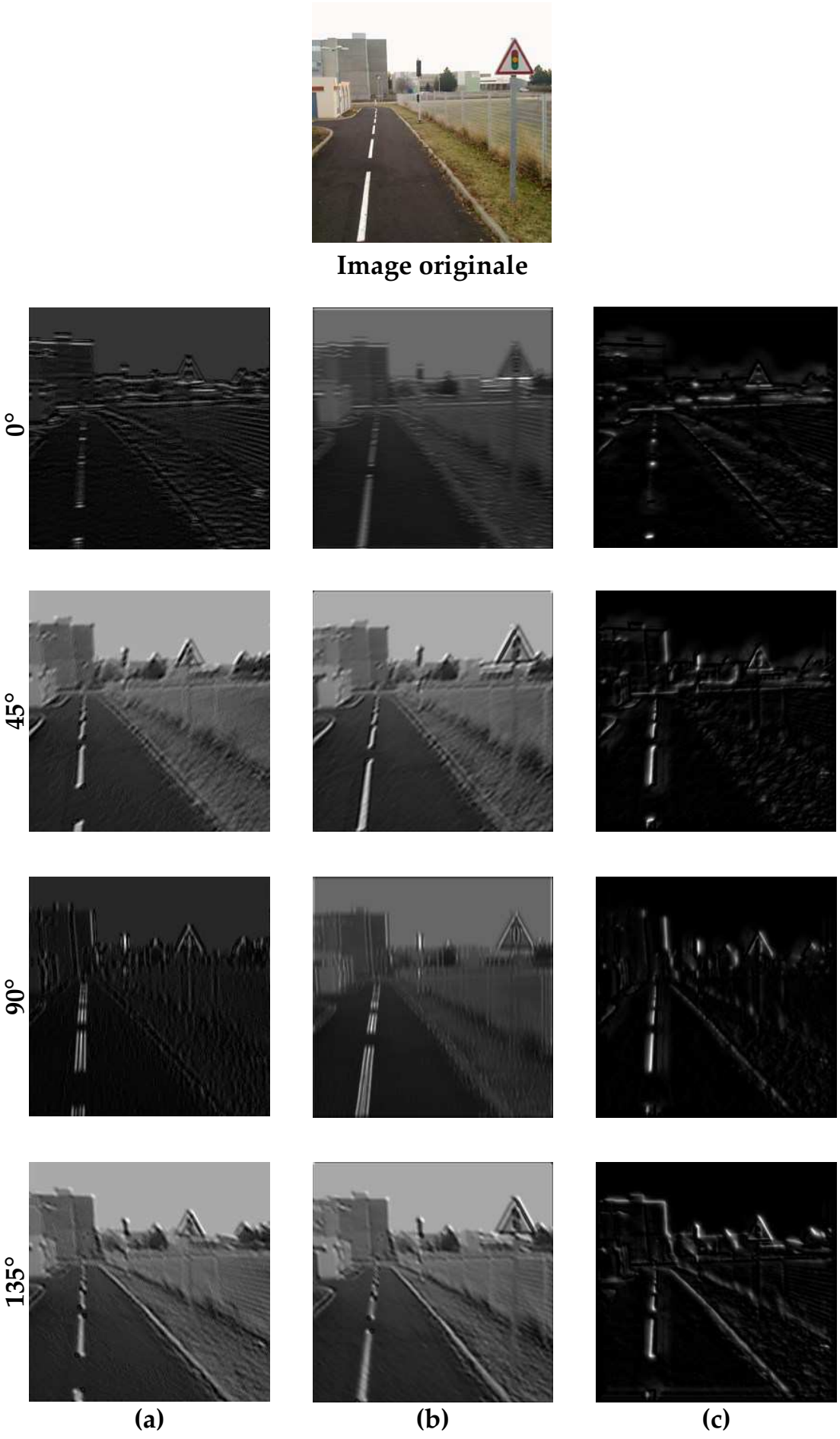


FIGURE 6.18: Les cartes d'évidences orientation (Image PAVIN).

Quatrième partie

CONCLUSION

"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."

"Ceci n'est pas la fin. Ce n'est même pas le commencement de la fin. Mais c'est, peut être, la fin du commencement."

Sir ***Winston Churchill*** (1874-1965), écrivain, historien et ancien premier ministre du Royaume-Uni. Discours prononcé en novembre 1942, après une victoire décisive des alliés contre l'Afrika Korps allemande en Égypte.

CONCLUSION ET PERSPECTIVES

PRINCIPALES CONTRIBUTIONS

Après une étude des primitives utilisées dans le domaine de la robotique mobile, nous avons proposé l'utilisation des régions saillantes pour la détection de fermeture de boucle et pour la cartographie topologique d'un environnement. Ce type de primitive est choisi principalement pour les raisons suivantes : un nombre réduit de primitives requises par image, une forte robustesse vis-à-vis des éventuelles perturbations et un taux de répétabilité élevé comparé à d'autres primitives.

Les deux approches *SAILMAP* et *Prob-SAILMAP* ont été présentées et utilisées sur des données réelles. Les résultats obtenus montrent que les primitives de type région saillante sont utilisables dans des applications de robotique et ainsi valident leur intérêt.

Pour palier à la complexité des algorithmes de détection de régions saillantes dans une image, nous avons utilisé la caméra intelligente *DreamCam* ayant comme unité de traitement FPGA. Le bon fonctionnement de ce système embarqué a été validé par l'implémentation d'algorithmes de détection de primitives, qui traitent les données sur le flot de pixels. Le premier était celui de Harris & Stephen pour la détection de point d'intérêt, algorithme auquel nous avons rajouté un module de description et de filtrage afin d'aboutir à un système d'extraction de points d'intérêt complet. Le second algorithme était le détecteur de régions saillantes, l'implémentation finale nous a permis d'obtenir un système fonctionnant en temps réel à une vitesse de 45 images/s en pleine résolution, c'est-à-dire 1280×960 pixels.

PERSPECTIVES

Quelques travaux restent cependant inachevés et seront poursuivis après la rédaction de ce manuscrit. En effet, les fonctions réalisant la pondération des différentes cartes et la description de régions saillantes n'ont pas encore été implémentées sur FPGA. C'est avec ces deux fonctions que l'on pourra aboutir à un système complet autonome pour la détection de régions saillantes sur une plateforme embarquée, que l'on pourra utiliser pour la détection de fermeture de boucle et la cartographie d'un environnement. D'autres travaux sont possibles pour améliorer notre système, comme par exemple la combinaison des deux détecteurs de primitives : régions saillantes et points d'intérêt, ou rajouter à l'algorithme de détection de régions saillantes d'autres primitives telles que : le mouvement, les formes, ... Pour notre implémentation sur FPGA, on peut prévoir la combinaison directe du filtre de dématriçage avec la chaîne de traitement du détecteur de régions saillantes. Une autre perspective ouverte est l'extension de l'architecture de détection de régions saillantes avec des modulations actives des traitements selon le contexte, en bénéficiant de l'évolution récente des capacités de reconfiguration dynamique des FPGA et des SoC-FPGA.

Quant à l'application de détection de fermeture de boucle, elle est déjà relativement stable et l'approche *SAILMAP* a donné de bons résultats en comparaison à FABMAP en environnement urbain. Enfin, pour rendre le système plus précis et plus efficace on peut rajouter un système d'activation et de désactivation de certaines cartes de primitive, suivant l'environnement exploré.

L'un des principaux avantages des régions saillantes est leur robustesse aux différents changements et perturbations qui peuvent arriver dans l'environnement. Afin de valider cette robustesse vis-à-vis du brouillard, nous prévoyons de faire des tests sur le site PAVIN avec un générateur de brouillard artificiel du **Cerema**[1] au sein du Département Laboratoire de Clermont-Ferrand (DLCF). La figure ci-dessous montre le dispositif complet utilisé pour la génération de brouillard via la plate-forme mobile, et donne un aperçu du résultat obtenu avec un tel dispositif.

(a) (b)

Le générateur de brouillard.

1 Le centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement ou Cerema est un établissement public à caractère administratif placé sous la tutelle conjointe des ministres chargés du développement durable, de l'urbanisme et des transports.

Cinquième partie

APPENDIX

PUBLICATIONS

Dans des revues internationales à comité de lecture :

- (*Accepté*) "DreamCam : A modular FPGA-Based Smart Camera Architecture", M. Birem & F. Berry, JSA-ELSEVIER, July 9th, 2012.
- (*Deuxième révision*) "PanoraMOS : A Panoramic Smart Camera", F. Pelissier, M. Birem & F. Berry, Journal of Signal Processing Systems, June 18th, 2014.

Dans les symposiums internationaux à comité de lecture :

- (*Accepté*) "FPGA-based Real time Extraction of visual features", M. Birem & F. Berry, IEEE International Symposium Circuits and Systems (ISCAS), May 20th-23th, 2012 | COEX, Seoul, [Korea].
- (*Accepté*) "Hardware Architecture for Visual Feature Extraction", M. Birem & F. Berry, IEEE/RSJ International Conference on Intelligent Robots and Systems, SCa-Bot'12 - First Workshop on Smart CAmeras for roBOTic applications , October 7th-12th, 2012 | Vilamoura, Algarve [Portugal]
- (*Accepté*) "Saliency in the FOG : Application to real and 2,5D pictures", R. Gana, JC. Quinton, M. Birem & al, TRA2014 Transport Research Arena, 14th-17th Apr 2014 Paris La Défense - France
- (*Accepté*) "SAIL-MAP : Loop-Closure Detection Using Saliency-Based Features", M. Birem, JC. Quinton, F. Berry, & Y. Mezouar, IROS 2014 Chicago, Illinois, Sept. 14th-18th, 2014.
- (*Soumis*) "Salient feature based localization using recursive Bayesian estimation", M. Birem, L. Lopez, JC. Quinton, F. Berry, & Y. Mezouar, ICRA 2015 Washington State Convention Center Seattle, Washington. May 26th - 30th, 2015.

A

LE DÉTECTEUR DE MORAVEC

Ce détecteur proposé par Moravec (1980) [132] est basé sur le calcul de SSD entre une fenêtre autour d'un pixel candidat et d'autres fenêtres décalées d'une faible distance dans un certain nombre de directions. Le calcul de la valeur d'intérêt est simplement le maximum d'un ensemble de valeurs E_i, chacune de ces valeurs se calcule par une somme de différence :

$$E(x,y) = \sum_{u,v} w(u,v)[I(x+u,y+v) - I(u,v)]^2 \tag{1.1}$$

où

- w spécifie la fenêtre/voisinage considérée (valeur 1 à l'intérieur de la fenêtre et 0 à l'extérieur),
- $I(u,v)$ est l'intensité au pixel (u,v),
- $E(x,y)$ représente une distance euclidienne entre la position initiale de la fenêtre et la seconde position après déplacement de (x,y).

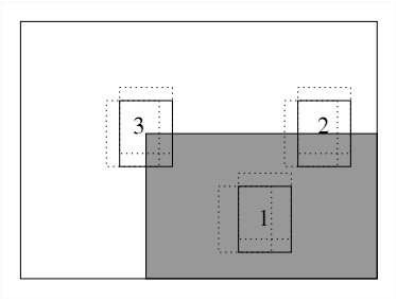


FIGURE 1.1: Les différentes situations considérées par le détecteur de Moravec.

Suivant la position du pixel considéré dans l'image, trois cas de figures sont envisageables (voir la Figure 1.1) :

1. Si le pixel est dans une zone homogène alors toutes les distances E_i calculées auront de faibles valeurs,
2. Si le pixel est proche d'un contour vertical ou horizontal, alors les déplacements perpendiculaires au contour auront des E_i de grandes valeurs par contre pour les déplacements parallèles ils auront de faibles valeurs,
3. Si le pixel considéré est un point d'intérêt ou à proximité d'un point d'intérêt alors tous les E_i auront de très grandes valeurs.

En conséquence, le principe du détecteur de *Moravec* est donc de rechercher les maximas locaux de la valeur minimale de E en chaque pixel. Cette valeur doit être supérieure au seuil prédéfini pour pouvoir dire que le pixel traité représente un point d'intérêt.

BIBLIOGRAPHIE

[1] *Altera Stratix-V*, 2014. URL http://www.altera.com/devices/fpga/stratix-fpgas/stratix-v/stxv-index.jsp.

[2] *Xilinx Virtex-7*, 2014. URL http://www.xilinx.com/products/silicon-devices/fpga/virtex-7/.

[3] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1597–1604. IEEE, 2009.

[4] Schmidt Adam, Kraft Marek, Fularz Michal, and Domaga Zuzanna. The comparison of point feature detectors and scriptors in the context of robot navigation. In *Workshop on Perception for Mobile Robots Autonomy*, 2012.

[5] L Albani, Pietro Chiesa, Daniele Covi, G Pedegani, A Sartori, and M Vatteroni. Visoc : A smart camera soc. In *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, pages 367–370. IEEE, 2002.

[6] Adrien Angeli, Stéphane Doncieux, Jean arcady Meyer, Université Pierre, Marie Curie Paris, and David Filliat. Real-time visual loop-closure detection. In *in IEEE International Conference on Robotics and Automation (ICRA*, pages 1842–1847, 2008.

[7] Kofi Appiah and Andrew Hunter. A single-chip fpga implementation of real-time adaptive background model. In Gordon J. Brebner, Samarjit Chakraborty, and Weng-Fai Wong, editors, *FPT*, pages 95–102. IEEE, 2005. ISBN 0-7803-9407-0. URL http://dblp.uni-trier.de/db/conf/fpt/fpt2005.html#AppiahH05.

[8] Clemens Arth, Horst Bischof, and Christian Leistner. Tricam-an embedded platform for remote traffic surveillance. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 125–125. IEEE, 2006.

[9] Hernán Badino, Daniel Huber, and Takeo Kanade. Real-time topometric localization. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1635–1642. IEEE, 2012.

[10] Sungmin Bae, Yong Cheol Peter Cho, Sungho Park, Kevin M Irick, Yongseok Jin, and Vijaykrishnan Narayanan. An fpga implementation of information theoretic visual-saliency system and its optimization. In *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, pages 41–48. IEEE, 2011.

[11] Dana H Ballard. Animate vision. *Artificial intelligence*, 48(1) :57–86, 1991.

[12] Shumeet Baluja and Dean A Pomerleau. Expectation-based selective attention for visual monitoring and control of a robot vehicle. *Robotics and autonomous systems*, 22(3) :329–344, 1997.

[13] Daniel Bauer, Ahmed Nabil Belbachir, Nikolaus Donath, Gerhard Gritsch, Bernhard Kohn, Martin Litzenberger, Christoph Posch, Peter Schön, and Stephan Schraml. Embedded vehicle speed estimation system using an asynchronous temporal contrast vision sensor. *EURASIP Journal on Embedded Systems*, 2007(1) :34–34, 2007.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[15] Bernard Bayle. Robotique mobile. *Ecole Nationale Superieure de Physique de StrasbourgUniversite Louis Pasteur*, 2008, 2007.

[16] Stéphane Bazeille and David Filliat. Incremental topo-metric slam using vision and robot odometry. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4067–4073. IEEE, 2011.

[17] Paul R Beaudet. Rotationally invariant image operators. In *Proceedings of the International Joint Conference on Pattern Recognition*, volume 579, pages 579–583, 1978.

[18] Richard Bellman, Richard Ernest Bellman, Richard Ernest Bellman, and Richard Ernest Bellman. *Adaptive control processes : a guided tour*, volume 4. Princeton University Press Princeton, 1961.

[19] Merwan BIREM and François BERRY. Fpga-based Real time Extraction of visual features. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 3053 –3056, may 2012. doi : 10.1109/ISCAS.2012.6271964.

[20] Erlei Bonato, Adriano K. Sanches, M. M. Fern, João M. P. Cardoso, E. D. V. Simoes, Eduardo Marques, São Carlos Brasil, and São Paulo. A real time gesture recognition system for mobile robots.

[21] Ali Borji, Majid Nili Ahmadabadi, Babak Nadjar Araabi, and Mandana Hamidi. Online learning of task-driven object-based visual attention control. *Image and Vision Computing*, 28(7) :1130–1145, 2010.

[22] Dimitris Bouris, Antonis Nikitakis, and Ioannis Papaefstathiou. Fast and efficient fpga-based feature detection employing the surf algorithm. In *Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, FCCM '10, pages 3–10, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4056-6. doi : 10.1109/FCCM.2010.11. URL http://dx.doi.org/10.1109/FCCM.2010.11.

[23] Ignacio Bravo, Javier Baliñas, Alfredo Gardel, José L Lázaro, Felipe Espinosa, and Jorge García. Efficient smart cmos camera based on fpgas oriented to embedded image processing. *Sensors*, 11(3) :2282–2303, 2011.

[24] Cynthia Breazeal and Brian Scassellati. A context-dependent attention system for a social robot. *rn*, 255 :3, 1999.

[25] Neil Bruce and MEd Jernigan. Evolutionary design of context-free attentional operators. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–429. IEEE, 2003.

[26] Neil Bruce and John Tsotsos. Saliency based on information maximization. In *Advances in neural information processing systems*, pages 155–162, 2005.

[27] Aurélie Bugeau. Attention visuelle multi-échelle. 2004.

[28] Antoni Burguera, Yolanda González, and Gabriel Oliver. Underwater slam with robocentric trajectory using a mechanically scanned imaging sonar. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3577–3582. IEEE, 2011.

[29] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4) :532–540, 1983.

[30] Nicholas J Butko and Javier R Movellan. Optimal scanning for faster object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2751–2758. IEEE, 2009.

[31] N. Petigara C. Shannon and S. Seshasai. A mathematical theory of communication. *Bell System Technical Journal*, 27(1) :379 – 423, January 1948. ISSN 1559-1662. doi : 10.1145/584091.584093. URL http://doi.acm.org/10.1145/584091.584093.

[32] Juan Gabriel Avin ? Cervantes. *Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi-structuré*. PhD thesis, Institut National Polytechnique de Toulouse, France, 2005.

[33] A Chauvin, J Herault, C Marendaz, and C Peyrin. Natural scene perception : visual attractors and images processing. *Progress in Neural Processing*, pages 236–248, 2002.

[34] Alan Chauvin. *Perception des scènes naturelles : étude et simulation du rôle de l'amplitude, de la phase et de la saillance dans la catégorisation et l'exploration de scènes naturelles*. PhD thesis, Grenoble 1, 2003.

[35] Eric Chown, Stephen Kaplan, and David Kortenkamp. Prototypes, location, and associative networks (plan) : Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1) :1–51, 1995. URL http://dblp.uni-trier.de/db/journals/cogsci/cogsci19.html#ChownKK95.

[36] Daesu Chung, Reid Hirata, T Nathan Mundhenk, Jen Ng, Rob J Peters, Eric Pichon, April Tsui, Tong Ventrice, Dirk Walther, Philip Williams, et al. A new robotics platform for neuromorphic vision : Beobots. In *Biologically Motivated Computer Vision*, pages 558–566. Springer, 2002.

[37] Dana Cobzas, Hong Zhang, and Martin Jagersand. Image-based localization with depth-enhanced image map. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1570–1575. IEEE, 2003.

[38] Jonathan Courbon, Youcef Mezouar, and Philippe Martinet. Autonomous navigation of vehicles from a visual memory using a generic camera model. *IEEE Transactions on Intelligent Transportation Systems*, 10(3) :392–402, 2009. URL http://dblp.uni-trier.de/db/journals/tits/tits10.html#CourbonMM09.

[39] Nicolas Courty and Eric Marchand. Visual perception based on salient features. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 1024–1029. IEEE, 2003.

[40] Philippe Coussy, Andres Takach, Michael McNamara, and Mike Meredith. An introduction to the systemc synthesis subset standard. In *Proceedings of the eighth*

IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pages 183–184. ACM, 2010.

[41] Mark Cummins and Paul Newman. Fab-map : Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27 (6) :647–665, 2008.

[42] Mark Cummins and Paul Newman. Highly scalable appearance-only slam-fab-map 2.0. In *Robotics : Science and Systems*, volume 1, pages 12–18. Seattle, USA, 2009.

[43] Andrew Dankers, Nick Barnes, and Alex Zelinsky. A reactive vision system : Active-dynamic saliency. In *Proc. of the 5th International Conference on Computer Vision Systems, ICVS, Bielefeld, Germany*, 2007.

[44] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

[45] Fabio Dias, François Berry, Jocelyn Sérot, and François Marmoiton. Hardware, design and implementation issues on a fpga-based smart camera. In *Distributed Smart Cameras, 2007. ICDSC'07. First ACM/IEEE International Conference on*, pages 20–26. IEEE, 2007.

[46] Patrick Dickinson, Kofi Appiah, Andrew Hunter, and Stephen Ormston. An fpga-based infant monitoring system. In Gordon J. Brebner, Samarjit Chakraborty, and Weng-Fai Wong, editors, *FPT*, pages 315–316. IEEE, 2005. ISBN 0-7803-9407-0. URL http://dblp.uni-trier.de/db/conf/fpt/fpt2005.html#DickinsonAHO05.

[47] Claas Diederichs. Fast visual servoing of multiple microrobots using an fpga-based smart camera system. In *Proc. of the 18th IFAC World Congress*, 2011.

[48] Benedikt Dietrich. *Design and Implementation of an FPGA-based Stereo Vision System for the EyeBot M6*. PhD thesis, University of Western Australia, 2009.

[49] Delphine Dufourd. Des cartes combinatoires pour la construction automatique de modèles d'environnement par un robot mobile. 2005.

[50] Magy Seif El-Nasr, Athanasios Vasilakos, Chinmay Rao, and Joseph Zupko. Dynamic intelligent lighting for directing visual attention in interactive 3-d scenes. *Computational Intelligence and AI in Games, IEEE Transactions on*, 1(2) :145–153, 2009.

[51] Hamed Fatemi. Processor architecture design for smart cameras. 2007.

[52] David Filliat. *Robotique Mobile*. ?cole Nationale Sup ?rieure de Techniques Avanc ?es ParisTech, 2013.

[53] David A Forsyth and Jean Ponce. *Computer vision : a modern approach*. Prentice Hall Professional Technical Reference, 2002.

[54] Silvia Franchini, Antonio Gentile, Filippo Sorbello, Giorgio Vassallo, and Salvatore Vitabile. An embedded, fpga-based computer graphics coprocessor with native geometric algebra support. *INTEGRATION, the VLSI journal*, 42(3) :346–355, 2009.

[55] Friedrich Fraundorfer, Christopher Engels, and David Nistér. Topological mapping, localization and navigation using image collections. In *IROS*, pages 3872–3877, 2007.

[56] Simone Frintrop. *VOCUS : A visual attention system for object detection and goal-directed search*, volume 2. Springer, 2006.

[57] Simone Frintrop. General object tracking with a component-based target descriptor. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4531–4536. IEEE, 2010.

[58] Simone Frintrop and Patric Jensfelt. Attentional landmarks and active gaze control for visual slam. *Robotics, IEEE Transactions on*, 24(5) :1054–1065, 2008.

[59] Gerald Fritz, Christin Seifert, Lucas Paletta, and Horst Bischof. Attentive object detection using an information theoretic saliency measure. In *Attention and Performance in Computational Vision*, pages 29–41. Springer, 2005.

[60] Dorian Galvez-Lopez and Juan D Tardos. Real-time loop detection with bags of binary words. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 51–58. IEEE, 2011.

[61] Dashan Gao and Nuno Vasconcelos. Discriminant saliency for visual recognition from cluttered scenes. In *NIPS*, 2004.

[62] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-aware saliency detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10) :1915–1926, 2012.

[63] Nuno Gracias and José Santos-Victor. Underwater video mosaics as visual navigation maps. *Computer Vision and Image Understanding*, 79(1) :66–91, 2000.

[64] Hayit Greenspan, Serge Belongie, Rodney Goodman, Pietro Perona, Subrata Rakshit, and Charles H Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 222–228. IEEE, 1994.

[65] Chenlei Guo and Liming Zhang. A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression. *Image Processing, IEEE Transactions on*, 19(1) :185–198, 2010.

[66] J-S Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pages 318–325. IEEE, 1999.

[67] Sunhyoung Han and Nuno Vasconcelos. Biologically plausible saliency mechanisms improve feedforward object recognition. *Vision research*, 50(22) :2295–2307, 2010.

[68] C Harris and M Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[69] Roger A Hart and Gary T Moore. *The development of spatial cognition : A review.* AldineTransaction, 1973.

[70] Gunther Heidemann, Robert Rae, Holger Bekel, Ingo Bax, and Helge Ritter. Integrating context-free and context-dependent attentional mechanisms for gestural object reference. In *Computer Vision Systems*, pages 22–33. Springer, 2003.

[71] J. M. Henderson, P. A. Weeks, and A. Hollingworth. The effects of semantic consistency on eye movements during complex scene viewing. *Journal of Experimental Psychology : Human Perception and Performance*, Vol 25(1), 1999.

[72] John M Henderson and Andrew Hollingworth. High-level scene perception. *Annual review of psychology*, 50(1) :243–271, 1999.

[73] Stephan Hengstler, Daniel Prashanth, Sufen Fong, and Hamid Aghajan. Mesheye : a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 360–369. ACM, 2007.

[74] Antonio Marin Hernandez. *Vision dynamique pour la navigation d'un robot mobile*. PhD thesis, Institut National Polytechnique de Toulouse, France, 2004.

[75] Kin Leong Ho and Paul Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3) :261–286, 2007.

[76] Byung-Woo Hong and Michael Brady. A topographic representation for mammogram segmentation. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2003*, pages 730–737. Springer, 2003.

[77] Xiaodi Hou and Liqing Zhang. Saliency detection : A spectral residual approach. In *CVPR*. IEEE Computer Society, 2007. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2007.html#HouZ07.

[78] L. Itti. *Models of Bottom-Up and Top-Down Visual Attention*. bu ;td ;mod ;psy ;cv, Pasadena, California, Jan 2000.

[79] Laurent Itti. Real-time high-performance attention focusing in outdoors color video streams. In *Electronic Imaging 2002*, pages 235–243. International Society for Optics and Photonics, 2002.

[80] Laurent Itti. The beobot platform for embedded real-time neuromorphic vision. *Advances in neural information processing systems*, 15, 2003.

[81] Laurent Itti. Automatic foveation for video compression using a neurobiological model of visual attention. *Image Processing, IEEE Transactions on*, 13(10) :1304–1318, 2004.

[82] Laurent Itti and Christof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision research*, 40(10) :1489–1506, 2000.

[83] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature reviews neuroscience*, 2(3) :194–203, 2001.

[84] Laurent Itti, Christof Koch, Ernst Niebur, et al. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11) :1254–1259, 1998.

[85] Laurent Itti, Christof Koch, and Jochen Braun. Revisiting spatial vision : Toward a unifying model. *JOSA A*, 17(11) :1899–1917, 2000.

[86] Matt Klein Jameel Hussein and Michael Hart. *Lowering Power at 28 nm with Xilinx 7 Series Devices*. Xilinx, January 2015.

[87] Seunghun Jin, Junguk Cho, Xuan Dai Pham, Kyoung Mu Lee, Sung-Kee Park, Munsang Kim, and Jae Wook Jeon. Fpga design and implementation of a real-time stereo vision system. *Circuits and Systems for Video Technology, IEEE Transactions on*, 20(1) :15–26, 2010.

[88] Matjaž Jogan and Aleš Leonardis. Panoramic eigenimages for spatial localisation. In *Computer Analysis of Images and Patterns*, pages 558–567. Springer, 1999.

[89] Michael Rubik Konrad J. Mayer Jorg Brodersen Johannes Furtler, Ernst Bodenstorfer and Christian Eckel. *High-Performance Smart Cameras*. Springer Science, 2010.

[90] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2) :83–105, 2001.

[91] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1) :35–45, 1960.

[92] Alonzo Kelly. Mobile robot localization from large-scale appearance mosaics. *The International Journal of Robotics Research*, 19(11) :1104–1125, 2000.

[93] Srinidhi Kestur, Dharav Dantara, and Vijaykrishnan Narayanan. Sharc : A streaming model for fpga accelerators and its application to saliency. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pages 1–6. IEEE, 2011.

[94] Kiyosumi Kidono, Jun Miura, and Yoshiaki Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. *Robotics and Autonomous Systems*, 40(2) :121–130, 2002.

[95] Kwanho Kim, Seungjin Lee, Joo-Young Kim, Minsu Kim, and Hoi-Jun Yoo. A configurable heterogeneous multicore architecture with cellular neural network for real-time object recognition. *IEEE Trans. Circuits Syst. Video Techn.*, 19(11) :1612–1622, 2009. URL http://dblp.uni-trier.de/db/journals/tcsv/tcsv19.html#KimLKKY09.

[96] Raymond Klein. Inhibitory tagging system facilitates visual search. *Nature*, 334 (6181) :430–431, 1988.

[97] Raymond M Klein and W Joseph MacInnes. Inhibition of return is a foraging facilitator in visual search. *Psychological science*, 10(4) :346–352, 1999.

[98] Christof Koch and Shimon Ullman. Shifts in selective visual attention : towards the underlying neural circuitry. In *Matters of Intelligence*, pages 115–141. Springer, 1987.

[99] Hemanth Korrapati. *Loop Closure for Topological Mapping and Navigation with Omnidirectional Images*. PhD thesis, Université Blaise Pascal, Clermont-Ferrand, France, 2013.

[100] Jana Košecká, Fayin Li, and Xialong Yang. Global localization and relative positioning based on scale-invariant keypoints. *Robotics and Autonomous Systems*, 52 (1) :27–38, 2005.

[101] Ben JA Kröse, Nikos Vlassis, Roland Bunschoten, and Yoichi Motomura. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, 19(6) :381–391, 2001.

[102] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1) : 191–233, 2000.

[103] Pierre-Jean Lapray, Dominique Ginhac, and Barthélémy Heyrman. HDR-ARtiSt : une caméra intelligente dédiée à la vidéo à grande dynamique en temps réel. In *24ème colloque Gretsi*, pages 1–4, Brest, France, September 2013. URL https://hal.archives-ouvertes.fr/hal-00932607.

[104] Fiack Laurent, Cuperlier Nicolas, and Miramond Benoît. Embedded and real-time architecture for bio-inspired vision-based robot navigation. *Real-Time Image Processing*, January 2014. doi : 10.1007/s11554-013-0391-9.

[105] Olivier Le Meur. *Attention sélective en visualisation d'images fixes et animées affichées sur écran : modèles et évaluation de performances-application*. PhD thesis, Nantes, 2005.

[106] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. *Acta Informatica*, 9 :23–29, 1977.

[107] Miriam Leeser, Shawn Miller, and Haiqian Yu. Smart camera based on reconfigurable hardware enables diverse real-time applications. In *FCCM*, pages 147–155. IEEE Computer Society, 2004. ISBN 0-7695-2230-0. URL http://dblp.uni-trier.de/db/conf/fccm/fccm2004.html#LeeserMY04.

[108] John Leonard and P Newman. Consistent, convergent, and constant-time slam. In *IJCAI*, pages 1143–1150, 2003.

[109] Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9) :1465–1479, 2006.

[110] Frédéric Lerasle. *Perception pour la robotique mobile en environnement humain*. PhD thesis, Université Paul Sabatier-Toulouse III, 2008.

[111] Shigang Li and Saburo Tsuji. Qualitative representation of scenes along route. *Image and vision computing*, 17(9) :685–700, 1999.

[112] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and Jan-Michael Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *Computer Vision–ECCV 2008*, pages 427–440. Springer, 2008.

[113] Huiying Liu, Shuqiang Jiang, Qingming Huang, and Changsheng Xu. A generic virtual content insertion system based on visual attention analysis. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 379–388. ACM, 2008.

[114] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2) :353–367, 2011.

[115] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[116] Jiebo Luo and Amit Singhal. On measuring low-level saliency in photographic images. In *2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)-Volume 1*, volume 1, page 1084. Published by the IEEE Computer Society, 2000.

[117] Vijay Mahadevan and Nuno Vasconcelos. Saliency-based discriminant tracking. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1007–1013. IEEE, 2009.

[118] Atsuto Maki, Peter Nordlund, and Jan-Olof Eklundh. Attentional scene segmentation : integrating depth and motion. *Computer Vision and Image Understanding*, 78(3) :351–373, 2000.

[119] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10) : 761–767, 2004.

[120] Emanuele Menegatti, Takeshi Maeda, and Hiroshi Ishiguro. Image-based memory for robot navigation using properties of omnidirectional images. *Robotics and Autonomous Systems*, 47(4) :251–267, 2004.

[121] Hongying Meng, Kofi Appiah, Andrew Hunter, and Patrick Dickinson. Fpga implementation of naive bayes classifier for visual object recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 123–128. IEEE, 2011.

[122] B Mertsching, M Bollmann, R Hoischen, and S Schmalz. The neural active vision system navis. *Handbook of Computer Vision and Applications*, 3 :543–568, 1999.

[123] François Berry Merwan Birem. Dreamcam : A modular fpga-based smart camera architecture. *Journal of Systems Architecture (JSA)*, 60 :519–527, 2014.

[124] François BERRY Merwan BIREM, Jean Charles QUINTON and Youcef MEZOUAR. Sailmap : Loop closure detection using saliency based features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[125] Jean-Charles Quinton François Berry Youcef Mezouar Merwan Birem, Léo Lopez. Salient feature based localization using recursive bayesian estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[126] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1) :63–86, 2004.

[127] Ruggero Milanese. *Detecting salient regions in an image : from biological evidence to computer implementation*. PhD thesis, University of Geneva, 1993.

[128] Ruggero Milanese, Jean-Marc Bost, and Thierry Pun. A bottom-up attention system for active vision. In *Proceedings of the 10th European conference on Artificial intelligence*, pages 808–810. John Wiley & Sons, Inc., 1992.

[129] Ajay K Mishra and Yiannis Aloimonos. Active segmentation. *International Journal of Humanoid Robotics*, 6(03) :361–386, 2009.

[130] Sara Mitri, Simone Frintrop, Kai Pervolz, Hartmut Surmann, and Andreas Nuchter. Robust object detection at regions of interest with an application in ball recognition. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 125–130. IEEE, 2005.

[131] TWJ Moorhead and T Binnie. Smart cmos camera for machine vision applications. In *IEE conference publication*, pages 865–869. Institution of Electrical Engineers, 1999.

[132] Hans Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*. September 1980.

[133] Romuald Mosqueron, Julien Dubois, and Michel Paindavoine. High-speed smart camera with high resolution. *EURASIP J. Emb. Sys.*, 2007, 2007. URL http://dblp.uni-trier.de/db/journals/ejes/ejes2007.html#MosqueronDP07.

[134] Claudia Muhl, Yukie Nagai, and Gerhard Sagerer. On constructing a communicative space in hri. In *KI 2007 : Advances in Artificial Intelligence*, pages 264–278. Springer, 2007.

[135] AC Murillo, P Campos, J Kosecka, and JJ Guerrero. Gist vocabularies in omnidirectional images for appearance based mapping and localization. *10th OMNIVIS, held with Robotics : Science and Systems (RSS)*, 3, 2010.

[136] Yukie Nagai. From bottom-up visual attention to robot action learning. In *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, pages 1–6. IEEE, 2009.

[137] Vinod Nair, Pierre-Olivier Laprise, and James J Clark. An fpga-based people detection system. *EURASIP Journal on Advances in Signal Processing*, (7) :1047–1061, 2005.

[138] Ken Nakayama and Manfred Mackeben. Sustained and transient components of focal visual attention. *Vision research*, 29(11) :1631–1647, 1989.

[139] Vidhya Navalpakkam and Laurent Itti. Modeling the influence of task on attention. *Vision research*, 45(2) :205–231, 2005.

[140] Vidhya Navalpakkam and Laurent Itti. An integrated model of top-down and bottom-up attention for optimizing detection speed. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2049–2056. IEEE, 2006.

[141] Marques Nicolas. *Méthodologie et architecture adaptative pour le placement e cace de tâches matérielles de tailles variables sur des partitions recongurables*. Theses, Université de Lorraine, November 2012. URL https://tel.archives-ouvertes.fr/tel-00823585.

[142] Alejandro Nieto, David López Vilarino, and Víctor Brea Sánchez. Towards the optimal hardware architecture for computer vision. pages 247–272, 2012. ISBN 978-953-51-0373-8.

[143] Ehsan Norouznezhad, Abbas Bigdeli, Adam Postula, and Brian C. Lovell. Object tracking on fpga-based smart cameras using local oriented energy and phase features. In *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*, ICDSC '10, pages 33–40, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0317-0. doi : 10.1145/1865987.1865993. URL http://doi.acm.org/10.1145/1865987.1865993.

[144] Navid Nourani-Vatani and Cedric Pradalier. Scene change detection for vision-based topological mapping and localization. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3792–3797. IEEE, 2010.

[145] Andress Nuchter, Hartmut Surmann, Kai Lingemann, Joachim Hertzberg, and Sebastian Thrun. 6d slam with an application in autonomous mine mapping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1998–2003. IEEE, 2004.

[146] Akihisa Ohya, Yukio Miyazaki, and Shin'ichi Yuta. Autonomous navigation of mobile robot based on teaching and playback using trinocular vision. In *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, volume 1, pages 398–403. IEEE, 2001.

[147] Aude Oliva and Antonio Torralba. Modeling the shape of the scene : A holistic representation of the spatial envelope. *International journal of computer vision*, 42 (3) :145–175, 2001.

[148] Aude Oliva and Antonio Torralba. Building the gist of a scene : The role of global image features in recognition. *Progress in brain research*, 155 :23–36, 2006.

[149] Aude Oliva, Antonio Torralba, Monica S Castelhano, and John M Henderson. Top-down control of visual attention in object detection. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–253. IEEE, 2003.

[150] Wilfried Osberger and Anthony J Maeder. Automatic identification of perceptually important regions in an image. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 701–704. IEEE, 1998.

[151] Nabil Ouerhani. *Visual attention : from bio-inspired modeling to real-time implementation*. Université de Neuchâtel, 2004.

[152] Lucas Paletta, Gerald Fritz, and Christin Seifert. Q-learning of sequential attention for visual object recognition from informative local descriptors. In *Proceedings of the 22nd international conference on Machine learning*, pages 649–656. ACM, 2005.

[153] N Parikh, L Itti, and J Weiland. Saliency-based image processing for retinal prostheses. *Journal of neural engineering*, 7(1) :016006, 2010.

[154] Sungho Park, Ahmed Al-Maashri, Yang Xiao, Kevin M. Irick, and Vijaykrishnan Narayanan. Saliency-driven dynamic configuration of hmax for energy-efficient multi-object recognition. In *ISVLSI*, pages 139–144. IEEE, 2013. URL http://dblp.uni-trier.de/db/conf/isvlsi/isvlsi2013.html#ParkAXIN13.

[155] Derrick Parkhurst, Klinton Law, and Ernst Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision research*, 42(1) :107–123, 2002.

[156] Frantz Pelissier and François Berry. Phd forum : Biseemos : A fast embedded stereo smart camera. In Richard P. Kleihorst, Andrea Prati, and Senem Velipasalar, editors, *ICDSC*, pages 1–3. IEEE, 2011. ISBN 978-1-4577-1708-6. URL http://dblp.uni-trier.de/db/conf/icdsc/icdsc2011.html#PelissierB11.

[157] Robert J. Peters and Laurent Itti. Applying computational tools to predict gaze direction in interactive visual environments. *ACM Trans. Appl. Percept.*, 5(2) :9 :1–9 :19, May 2008. ISSN 1544-3558. doi : 10.1145/1279920.1279923. URL http://doi.acm.org/10.1145/1279920.1279923.

[158] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*. IEEE Computer Society, 2007. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2007.html#PhilbinCISZ07.

[159] Vikas Reddy, Conrad Sanderson, Brian C Lovell, and Abbas Bigdeli. An efficient background estimation algorithm for embedded smart cameras. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–7. IEEE, 2009.

[160] Taha Ridene and Antoine Manzanera. Mécanismes d'attention visuelle sur rétine programmable. *Traitement et Analyse de l'Information : Méthodes et Applications (TAIMA'07)*, pages 301–306, 2007.

[161] Ruth Rosenholtz, Amal Dorai, and Rosalind Freeman. Do predictions of visual perception aid design ? *ACM Transactions on Applied Perception (TAP)*, 8(2) :12, 2011.

[162] Paul L Rosin. A simple method for detecting salient regions. *Pattern Recognition*, 42(11) :2363–2371, 2009.

[163] Eric Royer. *Cartographie 3D et localisation par vision monoculaire pour la navignation autonome d'un robot mobile*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2006.

[164] Albert Ali Salah, Ethem Alpaydin, and Lale Akarun. A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3) :420–425, 2002.

[165] MA-M Salem, K Klaus, F Winkler, and B Meffert. Resolution mosaic-based smart camera for video surveillance. In *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, pages 1–7. IEEE, 2009.

[166] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[167] Christian Scheier and Steffen Egner. Visual attention in a mobile robot. In *Industrial Electronics, 1997. ISIE'97., Proceedings of the IEEE International Symposium on*, volume 1, pages SS48–SS52. IEEE, 1997.

[168] Hossein Shahbazi and Hong Zhang. Application of locality sensitive hashing to realtime loop closure detection. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1228–1233. IEEE, 2011.

[169] Christian Siagian and Laurent Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2) :300–312, 2007.

[170] Christian Siagian and Laurent Itti. Biologically inspired mobile robot vision localization. *Robotics, IEEE Transactions on*, 25(4) :861–873, 2009.

[171] Chanop Silpa-Anan and Richard Hartley. Localisation using an image-map. In *Proceedings of the Australian conference on robotics and automation*. Citeseer, 2004.

[172] Stephen M Smith and J Michael Brady. Susan - a new approach to low level image processing. *International journal of computer vision*, 23(1) :45–78, 1997.

[173] See Fast Technologies. *Caméra rapide intelligente ProcImage 250*. Par Pereire - Bat B 99 Rue Pereire 78100 St-Germain-en-Laye, 2014.

[174] See Fast Technologies. *Caméra rapide intelligente ProcImage500-Eagle*. Par Pereire - Bat B 99 Rue Pereire 78100 St-Germain-en-Laye, 2014.

[175] Timothy B. Terriberry, Lindley M. French, and John Helmsen. Gpu accelerating speeded-up robust features, 2008.

[176] Sebastian Thrun et al. Robotic mapping : A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[177] Antonio Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[178] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and vision computing*, 16(2) :75–87, 1998.

[179] Anne Treisman and Stephen Gormican. Feature analysis in early vision : evidence from search asymmetries. *Psychological review*, 95(1) :15, 1988.

[180] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1) :97–136, 1980.

[181] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors : a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3) :177–280, 2008.

[182] Tinne Tuytelaars and Luc J Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC*, volume 412, 2000.

[183] Tommi Tykkala and Andrew I Comport. A dense structure model for image based stereo slam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1758–1763. IEEE, 2011.

[184] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1023–1029. Ieee, 2000.

[185] Christoffer Valgren, Tom Duckett, and Achim Lilienthal. Incremental spectral clustering and its application to topological mapping. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4283–4288. IEEE, 2007.

[186] Jan Van Der Horst, Rien Van Leeuwen, Harry Broers, Richard Kleihorst, and Pieter Jonker. A real-time stereo smartcam, using fpga, simd and vliw. In *Proc. 2nd Workshop on Applications of Computer Vision (Graz, May 12), Austria*, pages 1–8, 2006.

[187] Sethu Vijayakumar, Jörg Conradt, Tomohiro Shibata, and Stefan Schaal. Overt visual attention for a humanoid robot. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 4, pages 2332–2337. IEEE, 2001.

[188] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2) :137–154, 2004.

[189] Dirk Walther and Christof Koch. Modeling attention to salient proto-objects. *Neural Networks*, 19(9) :1395–1407, 2006.

[190] S. H. Wang. Saliency detection on fpga using accelerators and evaluation of algorithmic skeletons. *Graduation Paper*, 2012.

[191] Charles C. Weems, Edward M. Riseman, Allen R. Hanson, and Azriel Rosenfeld. The darpa image understanding benchmark for parallel computers. *J. Parallel Distrib. Comput.*, 11(1) :1–24, 1991. URL http://dblp.uni-trier.de/db/journals/jpdc/jpdc11.html#WeemsRHR91.

[192] Zhaoyi Wei, Dah-Jye Lee, Brent Nelson, and Michael Martineau. A fast and accurate tensor-based optical flow algorithm implemented in fpga. In *Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on*, pages 18–18. IEEE, 2007.

[193] James William. *The principles of psychology*. 1890.

[194] Wayne Wolf, Burak Ozer, and Tiehan Lv. Smart cameras as embedded systems. *Computer*, 35(9) :48–53, 2002.

[195] Jeremy M Wolfe and Todd S Horowitz. What attributes guide the deployment of visual attention and how do they do it ? *Nature Reviews Neuroscience*, 5(6) :495–501, 2004.

[196] Tingting Xu, Thomas Pototschnig, Kolja Kühnlenz, and Martin Buss. A high-speed multi-gpu implementation of bottom-up attention using cuda. In *ICRA*, pages 41–47. IEEE, 2009. URL http://dblp.uni-trier.de/db/conf/icra/icra2009.html#XuPKB09.

[197] Steven Yantis and John Jonides. Abrupt visual onsets and selective attention : Voluntary versus automatic allocation. *Journal of Experimental Psychology : Human Perception and Performance*, pages 121–134, 1990.

[198] Alfred L Yarbus, Basil Haigh, and Lorrin A Rigss. *Eye movements and vision*, volume 2. Plenum press New York, 1967.

[199] H. Yasuura, N. Takagi, and S. Yajima. The parallel enumeration sorting scheme for vlsi. *IEEE Trans. Comput.*, 31 :1192–1201, December 1982. ISSN 0018-9340. doi : http://dx.doi.org/10.1109/TC.1982.1675943. URL http://dx.doi.org/10.1109/TC.1982.1675943.

[200] Li Yiran. Fpga implementation for image processing algorithms. *Digital Signal Processing*, December 2006.

[201] Haoming Zeng. Fpga based smart nir camera. Master's thesis, Mid Sweden University The Department of Information Technology and Media (ITM), 2012.

[202] Hong Zhang, Bo Li, and Dan Yang. Keyframe detection for appearance-based visual slam. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2071–2076. IEEE, 2010.

[203] Nan Zhang. Computing optimised parallel speeded-up robust features (P-SURF) on multi-core processors. *International Journal of Parallel Programming*, 38(2) :138–158, 2010. doi : 10.1007/s10766-009-0122-9. URL http://dx.doi.org/10.1007/s10766-009-0122-9.

[204] Zoran Zivkovic, Bram Bakker, and Ben Krose. Hierarchical map building using visual landmarks and geometric constraints. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2480–2485. IEEE, 2005.

[205] Zoran Zivkovic, Olaf Booij, and Ben Kröse. From images to rooms. *Robotics and Autonomous Systems*, 55(5) :411–418, 2007.