# Détection et analyse du mouvement sur système de vision à base de rétine numérique

Julien Richefeu

THESE de DOCTORAT de l'UNIVERSITE PARIS 6

Spécialité :
Informatique

présentée par

M. Julien Richefeu

pour obtenir le grade de
DOCTEUR de l'UNIVERSITE PARIS 6

Sujet de la thèse :

# Détection et analyse du mouvement sur système de vision à base de rétine numérique

préparée au
Laboratoire d'Electronique et d'Informatique
Ecole Nationale Supérieure de Techniques Avancées
32, boulevard Victor
75739 Paris Cedex 15

Soutenue le 14 Décembre 2006
devant le jury composé de :

| | |
|---|---|
| M. Jean Louchet (INRIA) | Directeur de Thèse |
| M. Patrick Bonnin (LISV) | Rapporteur |
| M. Michel Paindavoine (LE2I) | Rapporteur |
| M. Patrick Garda (LIS) | Examinateur |
| M. Antoine Manzanera (LEI/ENSTA) | Examinateur |
| M. Bertrand Zavidovique (IEF) | Examinateur |
| M. Nicolas Lomenie (URD-P5) | Invité |

Détection et analyse du mouvement sur système de vision à base de rétine numérique.

**Mots-clés**

| | |
|---|---|
| Vision artificielle | Algorithmique |
| Détection du mouvement | Morphologie mathématique |
| Rétine programmable | Points d'intérêt |
| Analyse du mouvement | Corrélation |

**Résumé**

La rétine numérique programmable est un imageur qui combine fonctions d'acquisition et de traitement de l'image au sein de chaque pixel. L'objectif de notre travail consiste à utiliser ce circuit dans un système de détection et d'analyse du mouvement en se conformant à ses capacités de calcul et de mémorisation limitées.

Nous présentons d'abord trois méthodes de détection du mouvement adaptées à nos contraintes : un calcul de fond par une moyenne récursive classique ; un estimateur statistique, le filtre $\Sigma$-$\Delta$ ; et un détecteur d'amplitude de variation, le filtrage morphologique oublieux temporel.

Puis, nous introduisons un opérateur de calcul de points d'intérêt qui nous permet de mettre en correspondance des pixels appartenant à des objets mobiles au cours du temps.

Nous soulignons enfin l'importance de l'élaboration d'une méthodologie de conception d'algorithmes en adéquation avec les particularités de la rétine artificielle.

Motion detection and analysis in digital retina-based vision systems.

**Keywords**

Artificial vision — Algorithmic

Motion detection — Mathematical morphology

Programmable retina — Interest points

Motion analysis — Correlation

**Abstract**

The programmable digital retina is an image sensor combining image acquisition and processing within each pixel. This work aims at building up a motion detection and analysis system based on this circuit while taking into account its limited memory and computing power.

We present three motion detection methods adapted to our constraints : first, a basic recursive mean-based background substraction ; then a statistical estimator, the Σ-Δ filter ; third, a variation amplitude detector, the forgetting morphological filter.

Then, we introduce an interest point detector which enables real-time mobile object pixel matching.

Through these examples we demonstrate the importance of the adquation of the algorithm design methodology with the specific features of the artificial retina.

# Remerciements

J'adresse mes remerciements à MM. Alain Sibille et Thierry Bernard, respectivement chef du Laboratoire d'Electronique et d'Informatique et directeur du groupe de recherche en algorithmique de vision et architecture de l'Ecole Nationale Supérieure de Techniques Avancées à Paris, pour m'avoir accueilli dans leur laboratoire.

Je remercie tout spécialement MM. Antoine Manzanera et Jean Louchet pour m'avoir encadré et pour leurs conseils inestimables.

J'exprime ma profonde reconnaissance à l'égard de tous les membres du jury. Je remercie en particulier Bertand Zavidovique de me faire l'honneur de présider ce jury.

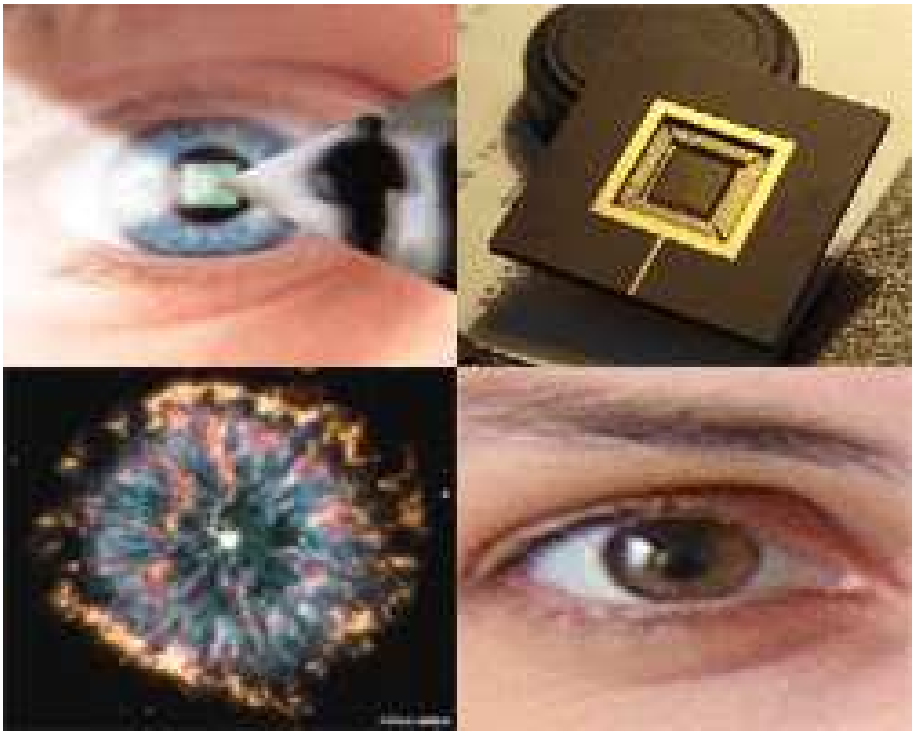
Conscient de la charge que cela représente, j'adresse mes profonds remerciements aux professeurs Michel Paindavoine et Patrick Bonnin pour avoir accepté d'être rapporteurs de ma thèse.

Je remercie également MM. Georges Stamon et Nicolas Loménie pour leurs précieux conseils et pour l'intérêt particulier qu'ils ont manisfesté tout au long du déroulement de ma thèse.

Je dédie ce manuscrit à ma femme pour ses encouragements et son soutien quotidien et sans faille.

Je salue tous mes amis, qui ont su m'oublier pour un temps...

Et bien sûr j'embrasse ma Pitchoune, Camille, qui aura mis moins de temps à voir le jour que ce mémoire ...



# Table des matières

# Table des figures

# Liste des tableaux

# Introduction générale

## Système de vision

Le système de vision de nombreux êtres vivants est l'aboutissement d'une longue évolution. Il permet de percevoir et d'interpréter une scène complexe contenant une multitude d'objets en un temps très court. Ce système de vision fait partie intégrante du système nerveux central.

Chez les primates par exemple, la rétine réalise l'acquisition et un premier niveau de traitement. L'image prétraitée est alors transmise au cerveau via les nerfs optiques. Notre cerveau est ainsi capable de résoudre les nombreux problèmes liés à la perception en temps réel sans le moindre effort apparent. L'architecture massivement parallèle de calcul collectif des neurones qui constituent notre système de perception visuel permet d'atteindre des performances importantes en termes de rapidité et de faible consommation énergétique.

L'ensemble *cornée* et *cristallin* (Figure 1) joue le rôle d'objectif qui permet de focaliser l'image, la courbure du cristallin étant réglable afin de permettre l'accommodation en fonction de la distance de l'objet. La *pupille* agit ainsi comme un diaphragme, elle permet l'adaptation du système visuel aux variations importantes d'éclairement. Le *cristallin* joue le rôle d'accommodateur des rayons, plaqué contre l'*iris*, il est suspendu par des fibres musculaires. La forme variable de cette lentille est contrôlée par des petits muscles. Bien que nous présentons brièvement l'optique occulaire dans son ensemble (nerf optique, chiasma, cortex visuel primaire, etc...), nous ne nous intéressons qu'aux traitements rétiniens.

Des études en neurophysiologie ont montré que les traitements de haut niveau réalisés par le système visuel humain ne sont possibles que grâce à un nombre important de prétraitements permettant d'extraire les paramètres multidimensionnels (la luminosité, la

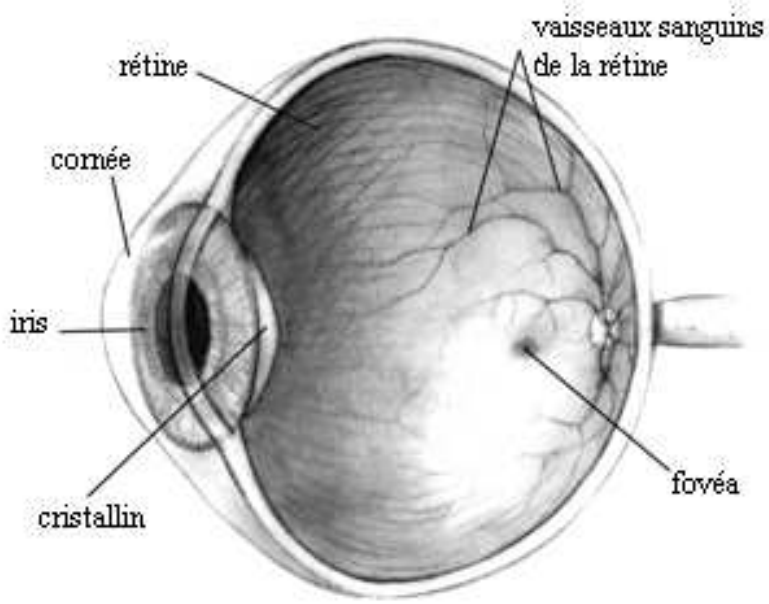



FIG. 1 – L'anatomie de l'œil (source wikipedia).

distance, le mouvement, etc...) de notre environnement.

La rétine (Figure 2) comporte plusieurs couches de cellules. La plus externe est celle des *photorécepteurs*. Viennent ensuite les *bipolaires*, qui forment avec les horizontales et les *amacrines* la couche des grains, et les *ganglionnaires*, dont les axones forment les fibres du nerf optique. L'organisation de la rétine est complexe : les connexions entre les diverses catégories de cellules et la nature des *neurotransmetteurs* libérés au niveau des *synapses* font l'objet actuellement d'études très poussées.

On classe les photorécepteurs en *cônes* et en *bâtonnets*, suivant la forme des structures qui portent le pigment photosensible. Il s'agit d'empilements *membranaires*, d'origine *ciliaire*[1] : dans le cas des bâtonnets, de nouvelles vésicules se forment à la base et le renouvellement est continu. Dans le cas des cônes, le renouvellement est plus complexe. La proportion relative des cônes, permettant la détection de la lumière en fort éclairement, et des bâtonnets, à seuil bien plus faible, qui assurent la vision en monochrome, varie d'un point à l'autre de la rétine. Chez l'homme, par exemple, la zone située sur l'axe optique

[1]Le corps ciliaire est la portion antérieure de la choroïde, sur lequel est attaché le cristallin, par l'intermédiaire des ligaments suspenseurs (ou zonules). Sur la face postérieure du corps ciliaire se trouvent les procès ciliaires qui sécrètent l'humeur aqueuse.

comporte davantage de cônes que sur sa périphérie, qui s'enrichit progressivement en bâtonnets. Il se forme ainsi une "tache jaune" due à la fois à l'absence de la couleur pourpre normale des bâtonnets et à la présence d'un pigment jaune. Fréquemment, cette même région centrale est déprimée en une fovéa[2], par écartement des neurones qui laissent ainsi plus facilement pénétrer les rayons lumineux.

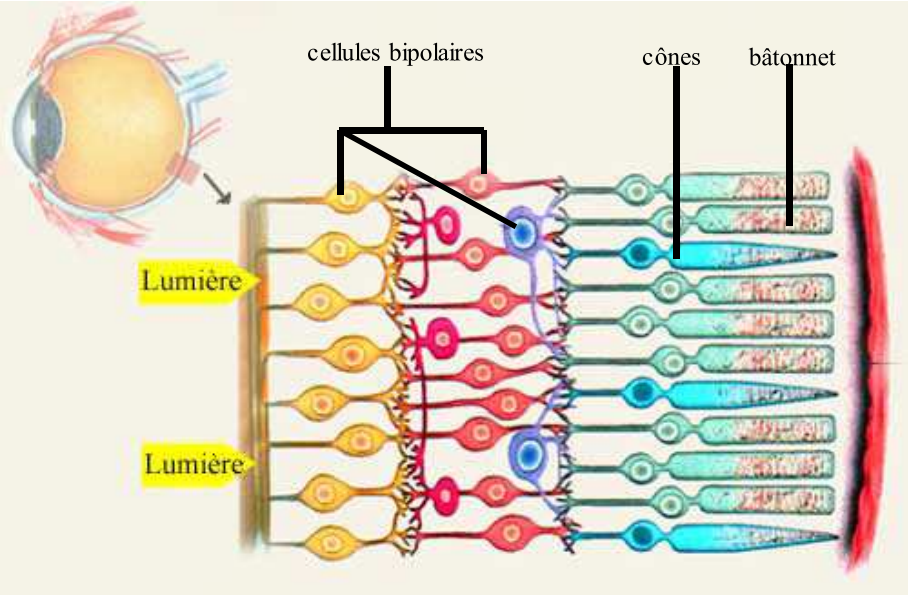



FIG. 2 – Coupe de la rétine d'un œil de vertébré (source wikipedia).

Chaque type de cône a une bande passante limitée, dans des longueurs d'ondes correspondant à ce que nous percevons comme des couleurs. Dans l'œil humain, il y a généralement trois types de cônes réagissant principalement aux couleurs rouge, vert et bleu. Des recherches actuelles[BKW98][Mj93] tendent à prouver que chez un certain pourcentage d'hommes (10%) et de femmes (50%), il existerait un quatrième type de cônes sensibles aux oranges. Les performances des bâtonnets, notamment en basse lumiére, nous permettent de voir dans l'obscurité presque complète mais seulement en noir et blanc. C'est pour cette raison que l'on ne voit pas les couleurs quand la lumière n'est pas assez forte, les performances des cônes étant beaucoup plus faibles.

La rétine constitue une première étape dans le processus de perception visuelle. Elle **réduit considérablement la quantité de données** à transmettre au cortex grâce aux prétraitements, locaux et globaux. Dans sa partie périphérique, la rétine humaine permet de **détecter les variations temporelles de luminosité** et la transformation de la

[2]La fovéa, la zone centrale de la macula, est la zone de la rétine où la vision des détails est la plus précise. Elle est située dans le prolongement de l'axe optique de l'œil

lumière incidente en signaux électriques analogiques, qui sont transmis au cortex visuel à travers le nerf optique. Par ailleurs, avant même les premiers traitements combinant plusieurs informations, un **codage logarithmique du flux lumineux** (*cônes*) intervient, bien adapté au caractère multiplicatif propre à la transmission et à la réflexion de la lumière. Cette propriété de compression nous permet de voir dans les conditions d'éclairement les plus diverses, de scènes en plein soleil à l'obscurité de la nuit (*bâtonnets*). Elle permet aux signaux de garder une dynamique raisonnable en comparaison à la dynamique de l'éclairement.

Cette brève présentation du système de vision humain nous permet de souligner les aptitudes principales que l'on cherche à reproduire dans un système de vision artificielle comme celui de la rétine numérique. Même sans chercher à mimer les fonctions visuelles biologiques, il convient qu'il existe un certains nombres d'analogies que nous allons montrer dans la suite de cette introduction. En effet, un **système de vision artificielle** se compose de différentes fonctions que l'on peut résumer en cinq niveaux principaux :

1. L'**acquisition** dont l'accès à l'image peut être aléatoire ou séquentiel ;
2. La **numérisation** qui peut être précédée de corrections (suppression du bruit spatial fixe par exemple) ;
3. Le **prétraitement** ou le **traitement bas niveau** qui permet une transformation de l'image au niveau du pixel (filtrage, segmentation,...) sans augmenter son niveau sémantique ;
4. Le **traitement moyen niveau ou intermédiaire** qui permet de remplacer l'image 2D en blocs structurés d'information. On augmente alors le niveau sémantique des données ;
5. Le **haut niveau ou niveau décisionnel** qui prend les informations issues du traitement de haut niveau et réalise une loi de commande (par exemple pour contrôler des actionneurs).

Les prétraitements, le niveau intermédiaire et décisionnel sont les trois niveaux classiquement établis de traitement en vision.

Jusqu'à la troisième étape, le flux de données représente souvent des séquences d'images, dont l'inconvénient majeur est la manipulation d'un volume important de données. De même, l'acquisition des images est une étape critique en terme de nombre d'opérations effectuées dans un système de vision. Le capteur d'image est une matrice de photodétecteurs[3] qui convertissent la lumière reçue en grandeur électrique après une durée d'exposition. Ces signaux sont, après une conversion en grandeurs numériques, traités pour extraire des informations plus utiles pour l'application.

[3]dispositifs électroniques qui transforment la lumière reçue par le capteur en signal utile (courant électrique)

Un système de vision artificielle conventionnel est composé d'un capteur d'image et d'un système de traitement de l'information ( microprocesseurs, DSP[4], FPGA [5], ... ). Les différentes fonctions sont séparées par le matériel utilisé : acquisition, prétraitement et traitement intermédiaire. Dans ce cas, il est impératif de convertir l'ensemble des données analogiques en données numériques, pour ensuite les traiter par les moyens numériques offerts par le circuit. Cette conversion constitue un goulet d'étranglement important[6] dû au flux massif des données lors des échanges entre le capteur et les divers modules de traitement.

La reproduction des mécanismes de calcul massivement parallèle[7] des systèmes de perception naturels a pour finalité de traiter l'information avant de la transmettre à un système d'analyse de plus haut niveau, réduisant ainsi la taille du flux de données.

La **rétine numérique programmable** fait partie de la famille des **capteurs intelligents**[8]. Ce sont des systèmes de perception qui disposent d'une certaine capacité de calcul assurée par des circuits analogiques ou numériques programmables permettant d'offrir plusieurs modes de fonctionnement, des fonctions de calibrage ou certaines corrections par exemple. Par rapport aux autres capteur intelligents qui permettent généralement de traiter les trois étapes fondamentales du traitement d'images, la rétine numérique comporte en son sein les deux étapes d'acquisition et de numérisation.

L'intérêt des traitements **embarqués**[9] de la rétine numérique est de tendre vers des applications **distribuées**[10], et de rendre **abstraits**[11] les détails des traitements sur les signaux bruts par rapport à l'information de plus haut niveau. La **communication**[12]

[4] *Digital Signal Processor* soit processeur de signal numérique, un composant électronique optimisé pour les calculs de traitement du signal.

[5] *Field-Programmable Gate Array*, un circuit intégré qui peut être reprogrammé après sa fabrication.

[6] au même titre que ceux dû au transfert d'informations entre des différentes couches du traitement

[7] comme le sont les réseaux neuronaux.

[8] *smart sensor* en anglais.

[9] Un système embarqué est un système intégré dans un système plus large avec lequel il est interfacé et pour lequel il réalise des fonctions particulières (contrôle, surveillance, communication). Les systèmes embarqués désignent aussi bien le matériel que le logiciel qui permettent de réaliser ces fonctions. Dans certaines applications, le matériel utilisé est un ordinateur, alors que dans d'autres, il s'agit de matériel dédié.

[10] Une application répartie ou distribuée consiste à répartir les fonctions d'une application sur plusieurs systèmes distincts.

[11] L'abstraction consiste à choisir, parmi l'ensemble des propriétés de plusieurs objets du monde réel ou imaginables, un certain nombre d'entre elles pour caractériser un objet-type, ou objet idéal, qui est ensuite plus commode à manier qu'une énumération d'objets réels, surtout si elle est infinie ...

[12] dans le cas de capteurs intelligents, on entend par communication la transmission vers et depuis les autres parties qui forme le système de vision numérique, on précisera dans la suite qu'elle peuvent être ces éléments

constitue éventuellement l'élément clé du concept de capteur intelligent puisqu'il est nécessaire de définir un **protocole**[13] sous forme d'opérateurs implantés sur le circuit du capteur, qui permet d'interagir avec le monde extérieur.

## Pourquoi s'intéresser au mouvement ?

Notre contexte de travail est celui de **capteurs fixe** (vidéosurveillance) **abandonnés**, donc permettant d'avoir une **longue autonomie** de travail sans intervention d'un opérateur. Il doit donc être capable de s'adapter á son environnement (**temporellement adaptatif**) et de ne nécessiter le moins possible de paramétrage (hormis éventuellement un réglage pré-opératoire). Il est important de souligner que le circuit rétinien PVLSAR34 utilisé lors de notre étude et les algorithmes que nous avons conçus et implémentés, font actuellement l'objet d'une utilisation dans le cadre d'un contrat industriel, pour des applications de vidéosurveillance dans le domaine militaire. Ainsi, bien que la détection du mouvement ne se limite pas à l'étude des cas où la caméra est fixe, pour être en adéquation avec notre cadre de travail, nous nous cantonnerons aux capteurs fixes ou animés de mouvement faibles pouvant se ramener au cas fixe.

Comme nous l'avons précédemment signalé, la rétine biologique est capable de détecter les variations temporelles de luminosité, c'est-à-dire qu'elle est en mesure de se focaliser sur certaines zones de la scène qui seront jugées intéressantes ou non ultérieurement. Ces zones correspondent notamment à l'information liée aux mouvements mais aussi à toute autre forme de variations significatives qui nécessitent un changement d'appréciation (ajustement de la pupille en fonction de la luminosité par exemple).

Afin de poursuivre notre étude comparative des visions artificielle et biologique, nous rappelons que la perception physiologique du mouvement comporterait en réalité trois phases principales :

1. **périphérique (préattentive)** : identification des régions dans le champ de vision dont les changements persistent entre deux instants (détection) ;
2. **attentive** : focalisation de l'attention sur ces parties afin de les observer en détail (estimation) ;
3. **cognitive** : liaison entre l'observation et la connaissance du monde (action).

La Figure 3 illustre un exemple typique de scène complexe à analyser dans notre contexte de travail. Cette image provient d'une séquence de trafic urbain où trois zones se

[13] un protocole de communication est une spécification de plusieurs règles pour un type de communication particulier

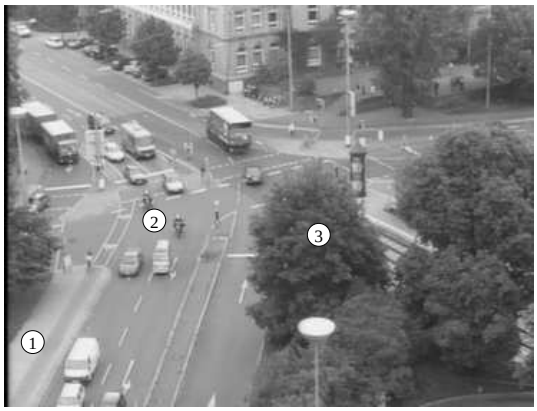



FIG. 3 – Exemple typique de scène complexe à analyser. La région (1) est une région du fond, sans mouvement, la région (2) est une zone de mouvement et la région (3) est une région de mouvement non pertinent, de bruit (source : Institut fur Algorithmen und Kognitive Systeme, Universität Karlsruhe, séquence kwbB, http ://i21www.ira.uka.de/image_sequences/).

distinguent aisément : une région de fond (1) sans aucun ou avec très peu de mouvement (un passage de piéton ou le vol d'un oiseau), une région de mouvement (2) correspondant au passage des véhicules et une région de bruit (3) due à des mouvements parasitaires (balancement des arbres sous l'effet du vent par exemple). L'objectif est de détecter avec une bonne précision spatiale les voitures puis de les suivre dans le temps (dans la scène). Il faut d'une part que le système ne détecte pas un oiseau ou les branches des arbres qui s'agitent au vent et d'autre part que le système soit suffisamment robuste pour suivre une voiture même lorsque celle-ci s'arrête à un feu rouge[14]. Tels sont les problèmes que nous allons aborder dans le cadre de ce travail.

## Contribution et structure de la thèse

Peu de travaux traitent du problème des rétines artificielles et encore moins nombreux sont ceux qui portent sur son utilisation dans un système de détection ou d'analyse du mouvement. Ceci explique qu'il nous a été difficile de concevoir un unique état de l'art regroupant tous les aspects de notre travail comme cela est souvent la cas dans les travaux de doctorat. Nous avons donc pris le parti de produire un état de l'art plus succinct à

[14]le suivi de cibles (*tracking* en anglais) s'éloigne de notre problématique et ne sera pas réellement aborder ici

chaque fois que le propos l'exige (notamment les sections traitant des capteurs intelligents, de la détection du mouvement, des points d'intérêt ou de l'estimation du mouvement).

En ce qui concerne le plan, ce manuscrit de thèse se compose en trois parties organisées par thématiques (capteurs intelligents, détection et analyse du mouvement et méthodologie de travail). Chaque thématique peut être prise indépendamment l'une de l'autre en fonction de l'intérêt que porte le lecteur sur tel ou tel problématique posée. Afin d'alléger la lecture, nous avons déporté une partie de la présentation des définitions de base, notamment celle de la morphologie mathématique, en annexe du recueil en s'y référant autant nécessaire.

Ainsi, en fonction du lecteur et de ses attentes, il est possible de parcourir le manuscrit d'un bout à l'autre de façon linéaire et en suivant le cheminement chronologique ou logique de notre travail ou bien de passer d'un chapitre à l'autre dans l'ordre qui convient le mieux au lecteur.

Notre travail de thèse a débuté par l'élaboration d'un lien entre détection et analyse du mouvement sous la forme d'un opérateur de calcul de points d'intérêt. En effet, nous avons cherché trés rapidement un moyen de suivre les structures dominantes issus de la phase de détection du mouvement. Ces structures se doivent de posséder une information trés concises et d'être facilement manipulable sur la rétine. Au vue des capacités de la rétine, le choix des points díntérêt s'est fait naturellement comme nous le montrons dans le Chapitre 4.

La Figure 4 présente les différents chemins de lecture possibles à travers les trois thématiques proposées. Ce schéma rappelle aussi les principales contributions que nous avons produites au cours de ce travail, les algorithmes $\Sigma - \Delta$ et de morphologie oublieuse temporelle pour la détection du mouvement et de calcul de points dominants principalement. Lors la lecture, ces productions seront encadrées afin d'en faciliter la visualisation.

La **première partie** (chapitre 1 et 2) s'intéresse à la problématique du traitement d'images sur un système à base de rétine numérique.

Le premier chapitre présente le concept de rétine artificielle, le circuit, ainsi que les différentes solutions proposées dans la littérature pour élaborer un système de vision embarqué.

Le second chapitre présente les différentes approches envisagées pour effectuer le traitement d'image sur rétine numérique. Nous y rappelons les principes fondamentaux de la morphologie mathématique et les avantages de son utilisation sur un circuit rétinien. Nous verrons par ailleurs les possibilités de traitements binaires puis en niveau de gris.

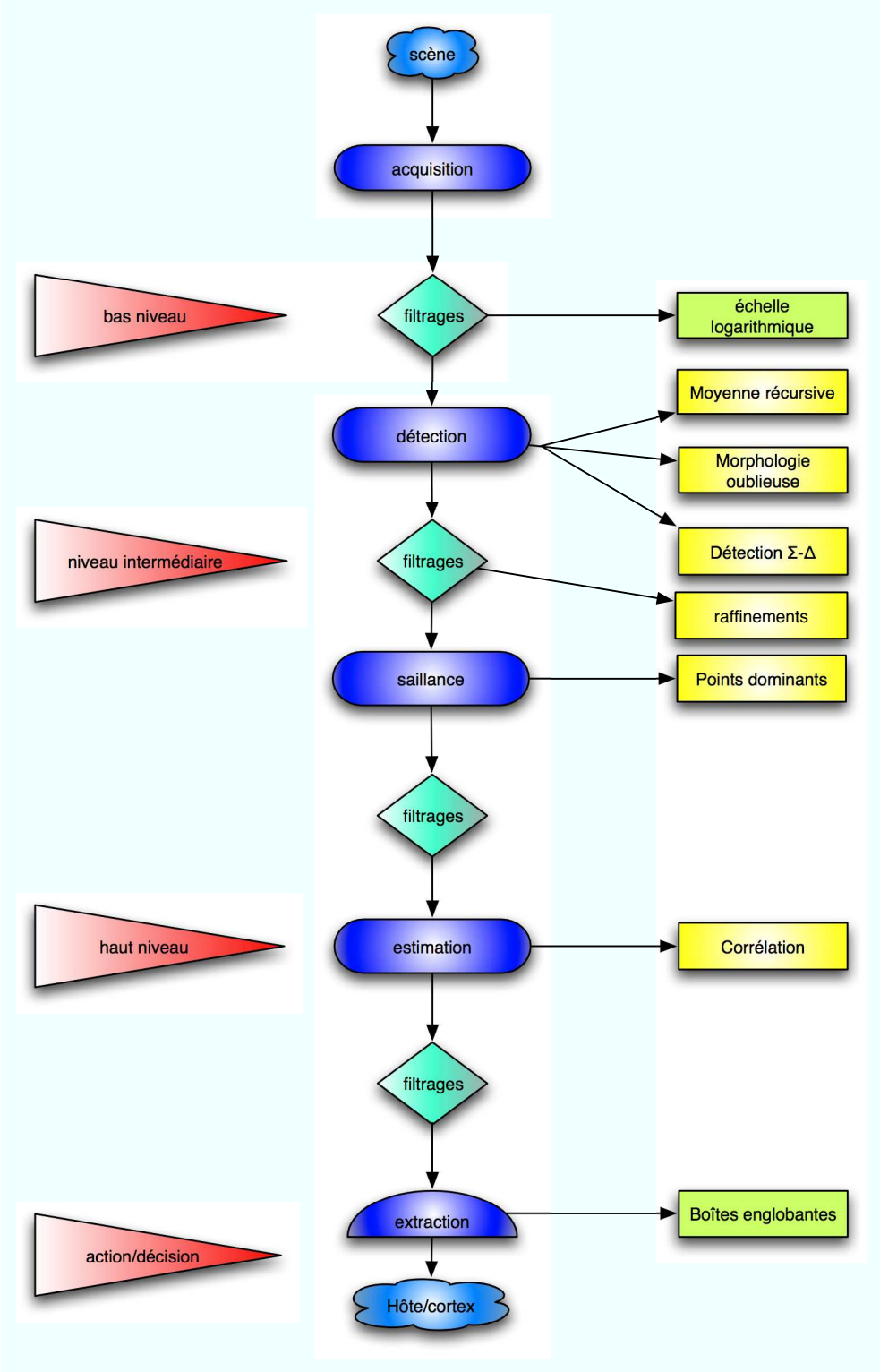



FIG. 4 – Schéma de lecture de la thèse.

La **deuxième partie** (chapitres 3, 4 et 5) explore la problématique de l'analyse du mouvement sur un système de vision rétinien.

Le troisième chapitre aborde un état de l'art des différentes techniques de détection du mouvement en fonction des hypothèses sur l'imageur. Nous présentons ensuite plusieurs techniques que nous avons implantées sur la rétine pour la détection. Nous évaluons ces différentes techniques avant de poursuivre sur les perspectives restant à explorer.

Le quatrième chapitre traite des structures saillantes spatiales que nous avons étudiées en tant que primitives du mouvement. Nous présentons diverses méthodes de calcul de squelettes morphologiques binaires puis en niveau de gris utilisées à la base d'un détecteur de points d'intérêt. Nous dressons ensuite un état de l'art des points d'intérêt avant de présenter la technique que nous avons employée sur la rétine artificielle pour détecter de tels points dominants.

Le chapitre cinq traite de l'estimation du mouvement servant comme complément à la détection ou de prétraitement en vue d'obtenir certaines primitives utiles, par exemple, pour le suivi de structures rigides dans la scène. Nous présentons à cette occasion un bref tour d'horizon des méthodes de corrélation et de calcul de flot optique.

La **troisième partie** (chapitre 6) présente les aspects méthodologiques à mettre en œuvre pour évaluer et valider des algorithmes de traitements d'images sur un système qui émule celui de la rétine numérique programmable. Cette étude propose en particulier des éléments de réflexion permettant la réalisation d'outils d'élaboration et de diagnostic d'algorithmes utilisant les primitives de calcul disponibles sur le circuit rétinien (méthodologie AAA[Sor94]).

Notre principale contribution est certainement la conception et l'implémentation[15] sur la rétine artificielle des algorithmes complets de détection du mouvement et de toute la chaîne méthodologique d'étude de ces algorithmes.

Au cours de cette étude, nous avons élaboré deux nouveaux opérateurs de détection du mouvement.

Le premier est un opérateur hybride, combinant filtres linéaires et morphologiques, nommé opérateur morphologique oublieux temporel en raison de sa particularité d'agréger les valeurs extrêmes au cours du temps tout en les oubliant progressivement.

[15]En ingénierie et plus particulièrement en informatique, l'implántation désigne la création d'un produit fini à partir d'un document de conception, d'un document de spécification, voire directement depuis un cahier des charges

Le second est un opérateur non linéaire basé sur une analogie avec une technique de conversion analogique / numérique en électronique, le filtre $\Sigma$-$\Delta$. Originellement employé pour approximer la médiane, il permet une détection précise et robuste des objets mobiles.

Nous avons aussi mis au point un algorithme de calcul de points d'intérêt basé sur l'utilisation de squelettes morphologiques pour le calcul des régions à forte courbure.

Enfin, nous nous sommes efforcés de maintenir des logiciels de validation de nos algorithmes et d'émulation de systèmes de vision à base de rétines numériques. Cela nous a amené à dégager les grandes lignes de ce que serait la conception d'algorithmes sur rétine artificielle et à l'abstraire en un ensemble d'outils de développement.

# Chapitre 1

# Système de vision rétinien

## 1.1 Introduction

Dans ce chapitre nous soulignons tout d'abord les avantages offerts par la rétine numérique programmable dans un système de vision artificielle, puis nous présentons un état de l'art synthétique des capteurs intelligents afin de situer le circuit sur lequel nous avons effectué nos recherches.

Nous ciblons ensuite notre propos sur la rétine artificielle, nous expliquons ses aspects techniques et nous rappelons les technologies qui ont permis son émergence tout en mettant l'accent sur les objectifs et les enjeux scientifiques de notre recherche. Nous présentons ainsi le contexte et les contraintes dans lequel se situe notre travail de recherche.

## 1.2 La rétine programmable

La rétine numérique programmable (ou rétine artificielle) est un **imageur**[1] **CMOS**[2] qui combine fonctions d'acquisition et de traitement de l'image au sein de chaque pixel. Un tel concept provient originellement d'une analogie avec le monde biologique où les êtres vivants traitent effectivement une partie de l'information provenant de la scène à

[1]Dispositif électronique permettant l'acquisition d'images (i.e. capteur d'image)

[2]Complementary Metal-Oxide-Semiconductor est un type de composant électronique à faible consommation électrique

l'aide des neurones tapissant le fond de leur rétine. C'est grâce aux récentes avancées dans le domaine de l'**intégration submicronique**[3] que la réalisation d'un tel système est possible.

L'un des avantages majeurs de ce concept, est la possibilité d'intégrer sur la même puce que le capteur, de nombreuses fonctions électroniques d'amplification, de conversion, de traitement et de séquencement. Il est ainsi possible de réaliser ainsi des systèmes de faible encombrement, très appréciés dans le cadre d'applications embarquées, présentant en outre une consommation réduite.

### 1.2.1 Intérêt du concept de rétine artificielle

Dans un système de traitement d'image conventionnel, l'image est acquise par un capteur bidimensionnel (caméra **CCD**[4] en général) puis transmise séquentiellement à un convertisseur analogique / numérique[5] et enfin à un ou plusieurs calculateurs (processeur FPGA, DSP[6], ...). Le flux d'informations en sortie du capteur représente un gros volume de données qu'il est nécessaire de traiter en temps réel[7].

Un capteur matriciel est composé d'éléments photosensibles appelés pixels, le plus souvent de forme carrée ou rectangulaire, répartis orthogonalement les uns par rapport aux autres de façon à réaliser $L$ lignes et $N$ colonnes. Chaque pixel constitue un point élémentaire d'échantillonnage de l'image. C'est au niveau de cet élément que se produit la conversion photons / électrons. L'information obtenue est soit un courant, soit une tension. Cette grandeur peut être exploitée de deux façons :

- une lecture analogique,
- une numérisation du signal en utilisant un convertisseur analogique / numérique. Dans ce cas, on obtient en sortie du système une information codée qui peut être visualisée ou mémorisée pour des traitements, au moyen d'un ordinateur, par exemple.

Le traitement à effectuer après la numérisation afin de prendre une décision ou d'effectuer une action se découpe en plusieurs étapes :

[3] pour exemple, la taille moyenne des motifs minimaux des microprocesseurs esr passée de 1 micron au début des années 1990 à moins de 0.05 micron aujourd'hui alors que pendant le même temps la fréquence d'horloge est passée de 66 Mhz à plus de 3 Ghz [Anc02].

[4] Charge-Coupled Device, dispositif à transfert de charges : désigne un capteur qui convertit la lumière en signaux électriques qui peuvent être ensuite numérisés

[5] CAN

[6] *Digital Signal Processor* soit processeur de signal numérique, un composant électronique optimisé pour les calculs de traitement du signal.

[7] Un système temps réel est contraint par le temps de calcul, la cadence vidéo est de 25 images par seconde soit 25Hz

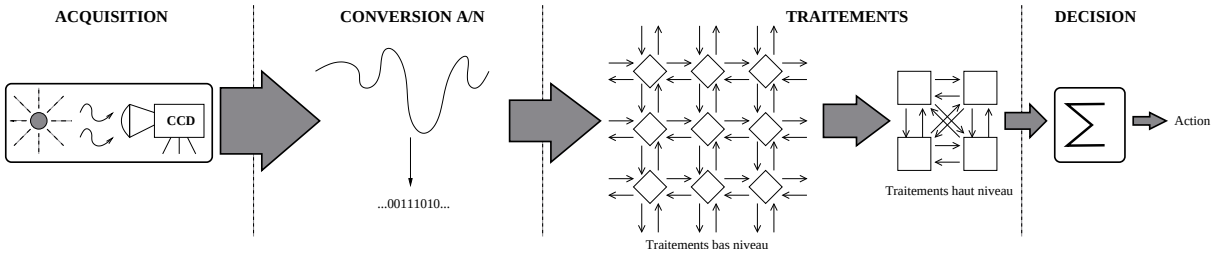



FIG. 1.1 – Approche classique d'un système de vision artificielle.

- des opérations de **bas niveau** qui traitent l'image localement, en chaque pixel. Cette étape se prête bien à une parallélisation massive (par exemple, le lissage pour enlever le bruit moyen, le médian gaussien).
- des opérations de **niveau intermédiaire** qui traitent des zones de l'image, des amas de pixels, des régions. Les données restent cependant structurées spatialement en distribution bidimensionnelle. Le volume d'informations à traiter se réduit en même temps que la possibilité d'effectuer des tâches en parallèle. Les données représentent des attributs tels que la surface par exemple.
- enfin des opérations de **décision** (haut niveau) qui agissent sur des données symboliques, de faible volume et dépourvues de structures spatiales (par exemple, on a reconnu ou non un élément de la scène).

La figure 1.1 présente le processus de traitement d'image sur une architecture classique en mettant en évidence leur faiblesse majeure. Les images qui sont sérialisées lors de l'extraction du capteur sont amenées à retrouver leur structure bidimensionnelle pour les traitements de bas niveau. Le flux d'information s'en trouve réduit et la dépense énergétique est souvent supérieure à celle réellement nécessaire au traitement.

Ces architectures standard atteignent une limite technologique pour des questions de coût, d'encombrement, de débit de données inhérentes à la chaîne d'acquisition (goulot d'étranglement sur les entrées/sorties) et à la puissance de calcul nécessaire.

Par opposition, dans un système à base de rétine artificielle, l'aquisition de l'image, la conversion analogique numérique et les traitements de bas niveaux sont réalisés au sein même du circuit de la rétine (Figure 1.2). Les données sont traitées en exploitant au maximum le parallélisme massif disponible au niveau de la matrice de capteurs. Les transferts entre circuits sont limités afin de réduire la consommation d'énergie. Les informations transmises en dehors du circuit rétinien sont des descripteurs d'image représentant un faible volume de données et facilement manipulables dans des algorithmes de plus haut niveau par des **processeurs scalaires**[8] hôtes associés qui complètent le système. De plus,

[8]Un processeur est dit scalaire s'il ne traite qu'une seule donnée à la fois

le processeur hôte, déchargé des traitements bas niveau, ne nécessite pas une grande puissance de calcul, ce qui permet d'exploiter des circuits compacts à faible consommation.

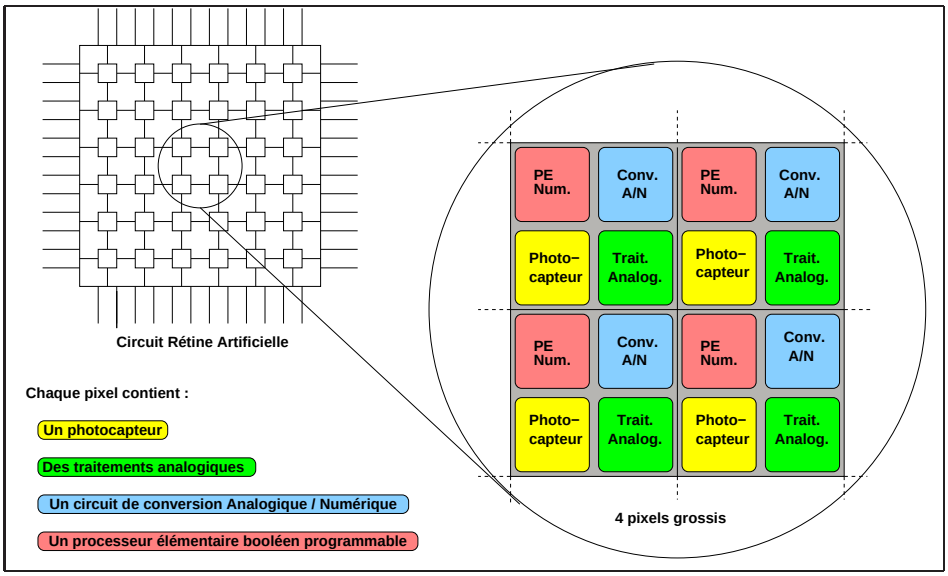



FIG. 1.2 – Illustration de l'architecture distribuée d'une rétine artificielle numérique programmable. Chaque pixel comporte un photocapteur, un élément de traitement analogique, un CAN et un processeur élémentaire numérique.

Ainsi, un système de vision rétinien répond à des exigences de performances en terme de :

– **vitesse** d'exécution, critique pour l'utilisation dans les systèmes temps réel.
Les rétines artificielles ont été inspirées par l'observation des rétines biologiques du système de vision humain. En effet, le système de vision humain, par exemple, est capable de percevoir et d'interpréter une multitude d'informations provenant d'une scène complexe en un temps très bref. Le travail de la rétine est donc de condenser l'information afin de fournir à son système externe (cortex) des images pré-traitées. Ces traitements de bas (ou moyen) niveau sont réalisés localement et en temps réel. De plus, ces traitements de données au sein du pixel évitent le goulet d'étranglement au niveau des entrées / sorties du capteur comme sur les systèmes de traitement d'images classiques.
Cependant, les rétines artificielles ne sont pas les seuls systèmes de traitement d'images capables de traiter l'information en temps réel. Une simple caméra associée à un processeur DSP, ou même à un ordinateur puissant permet d'effectuer ces mêmes opérations de traitements d'image en temps réel. La vitesse de traitement n'est donc pas le seul enjeu majeur des rétines artificielles.
Bien sr, le temps de calcul dépend aussi du type de traitement effectué et des

résolutions utilisées pour l'image à traiter. Certains types de calculs se prêtent mieux, comme nous le verrons, au parrallélisme massif que d'autres. En ce qui concerne la taille des images, un circuit massivement parralèle n'a normalement pas de temps de calcul dépendant de la résolution hormis pour certains calcul (les calculs régionnaux en règle général, que nous n'aborderons pas ici).

– **compacité**, requise pour les applications dans les systèmes embarqués.
Bien que la solution d'intégrer des éléments de calcul et de mémorisation au sein de chaque pixel paraisse idéale, les techniques d'intégration CMOS ne la rendent possible que depuis quelques années.
– **consommation d'énergie**, primordiale pour les systèmes autonomes.
Elle permet d'éviter des transferts massifs de données sur des longues distances, transferts qui se montrent très pénalisants en vitesse de traitement mais surtout en consommation d'énergie en raison des capacités des bus de transmission.
Si l'on considère maintenant la conjonction entre la vitesse de traitement élevée, la compacité du circuit et la faible consommation d'énergie, alors les systèmes concurrentiels aux rétines artificielles comme ceux évoqués ci-dessus sont beaucoup moins compétitifs.

Les capteurs numériques atteignent aujourd'hui une certaine fiabilité et sont utilisés dans le secteur grand public (caméscopes, photographies numérique, webcam, cellule photographique de téléphones portables, ...), le secteur automobile (anti-collision, guidage, détecteur d'obstacles, ...) et des applications demandant des vitesses de lecture particulièrement élevées (test de crashs, analyse d'explosion) ainsi que la vidéosurveillance, la défense, ou encore pour des applications en imagerie médicale.

Afin d'améliorer l'élaboration de systèmes de vision à base de rétines numériques et de proposer des solutions optimales pour ces applications, il nous faut adapter les méthodes existantes et repenser une partie de l'algorithmie ( voir Adéquation Algorithme Architecture [Sor94]). Telle est la principale caractéristique de notre travail. Nous commençons par nous situer dans les recherches existantes en effectuant un tour d'horizon historique des capteurs intelligents.

### 1.2.2 Les capteurs intelligents

Les premiers travaux visant à reproduire le comportement des rétines biologiques à l'aide de composants électroniques ont été menés dans le monde de la recherche biologique. La découverte des algorithmes de traitement de l'information qui, partant d'une image, permettent de créer une "traduction" de la scène utile pour la perception visuelle relève en effet d'une inspiration souvent biologique. Ainsi, au milieu des années 60, l'équipe de Runge [RUV68] tente de reproduire une rétine de pigeon avec des composants discrets.

L'idée ést d'effectuer les traitements de bas niveau le plus proche possible du pixel, donc du photorécepteur.

L'équipe de Fukushima [FYYN70] dans les années 70 met en place un réseau analogique de 700 photorécepteurs. Et à la fin des années 70, les progrès technologiques et l'avènement du silicium permettent l'intégration à très grande échelle de composants. Le début des années 80 voit la prolifération des circuits neuronaux mais le couple processeur-photorécepteur reste rare, mise à part l'élaboration d'une souris informatique optique [Lyo81]. Dès lors on distingue principalement trois équipes universitaires qui travaillent sur les rétines avec des approches différentes : l'Institut Technologique de Californie (CalTech, USA) [MC80], l'Université de Linköping en Suède [FO83] et l'ETCA avec l'Université Paris Sud XI Orsay en France [Zav81] [Gar84] [DGZ85] [DO96].

Le début des années 90 voit l'apogée des **RNA**[9] qui sont peu à peu abandonnés par faute de retombées industrielles conséquentes, hormis le succès prometteur du Trackball Marble de Logitech, et de quelques systèmes de reconnaissance d'écriture. Cet échec profite aux capteurs intelligents actuels, concurrents des RNA, à base de CMOS qui bénéficient des progrès technologiques liés à la finesse de gravure des microprocesseurs. Nos recherches quant à elles s'appuient sur les nombreuses études architecturales [Gar84] [Pai01] [Ber92] et algorithmiques [Man00] menées sur les systèmes de vision à base de rétine numérique artificielle programmable.

Les travaux de ces quarante dernières années se sont développés dans des approches différentes dont nous expliquons maintenant la teneur. Plusieurs types de capteurs intègrent en effet les fonctionnalités de calcul au bord de la matrice imageuse, dans le plan focal. Les rétines numériques et certaines rétines analogiques[10] cherchent à effectuer les calculs directement dans le pixel. Dans les approches voulant reproduire le comportement biologique des rétines animales, la conception du circuit est entièrement fondée sur une tentative de reproduction en silicium de fonctions observées dans le monde animal, principalement chez les insectes (perception du mouvement, inhibitions latérales, vision périphérique, capteur fovéal).

Cependant, la définition du concept rétinien n'est pas la même pour toutes les équipes de recherche travaillant dans le domaine des capteurs de vision. Alors que certaines équipes tendant à appeler "Rétine" tout capteur de type CMOS, d'autres équipes définissent les rétines artificielles comme des imageurs comportant une unité de traitement de l'information au sein de chaque pixel (*intelligence dans le pixel*). Ces unités peuvent faire appel à des montages soit analogiques, soit numériques, soit aux deux à la fois, appelés **proces-**

[9] Réseaux de Neurones Analogiques

[10] des circuits électroniques comportant des éléments de calculs analogiques, voir ci-après

**seurs élémentaires** (PE).

### Les circuits analogiques dédiés

A la fin des années 80, les rétines artificielles analogiques se sont établies en tant que domaine de recherche international [Mea89][Del83] (détection des discontinuités spatiales et temporelles [DeW92][KMaJH+93], détection des contours [AB96], détection et analyse du mouvement [HKLM88][Moi97a][Xa96][TH99a], stéréovision [Mah94], ...) dans des laboratoires aussi prestigieux que le CalTech[11], le MIT[12] et l'Université John Hopkins aux USA, l'Université d'Adelaïde en Australie, le Centre Suisse d'Electronique et de Microtechnique de Neuchatel, l'Université de Séville en Espagne, et l'IEF[13] de l'Université Paris XI Orsay et l'INPG[14] en France.

La **neurophysiologie**[15] de la vision animale, par exemple, a constitué une source d'inspiration importante pour la majeure partie des conceptions de circuits dédiés qui ont été publiées. Ces types de circuits exploitent les propriétés physiques du transistor CMOS qui le compose, offrant une grande richesse en utilisant ses différents modes (faible/forte inversion). On obtient alors un excellent rapport vitesse sur consommation par surface.

Il y a cependant des limitations importantes au traitement analogique de l'information dans le pixel. D'une part, deux transistors géométriquement identiques sur un même circuit intégré n'ont pas exactement les mêmes caractéristiques de fonctionnement (même s'ils sont voisins) en raison des inhomogénéités du processus technologique de fabrication. Cette particularité génère un bruit dit spatial fixe sur l'ensemble de la matrice (cela est aussi vrai en numérique mais est moins critique que pour les circuits analogiques car ceux-ci nécessitent des mesures plus fines). D'autre part, avec des courants permanents dans chaque pixel, les contraintes relatives à la consommation de puissance incitent à utiliser les transistors en faible inversion[16]. Dans ces conditions, il faut trouver des solutions spécifiques aussi bien au niveau du circuit qu'au niveau du système de traitement, avec un risque de conception plus ou moins important.

Pour la plupart, les opérateurs analogiques pour la vision artificielle semblent avoir une structure assez simple par rapport aux opérations qu'ils sont capables de mener. Cela

[11] California Institute of Technology
[12] Massachussets Institute of Technology
[13] Institut d'Electronique Fondamentale
[14] Institut Nationale Polytechnique de Grenoble
[15] La neurophysiologie est l'étude des fonctions du système nerveux, reposant sur tous les niveaux de description, du niveau moléculaire jusqu'au niveau le plus intégré des réseaux neuronaux.
[16] un mode de fonctionnement où le bruit spatial fixe peut être aussi important que le signal lui-même

provient surtout de la forte régularité des calculs implantés. Par exemple, des fonctions qui décroissent avec la distance ou avec le temps se prêtent assez naturellement à l'exploitation de transistors sous la forme de composants passifs, permettant des implantations très compactes. En effet, moins de régularité se traduit généralement par une hausse sensible de la consommation en surface. Une autre source de consommation de surface réside dans les spécificités fonctionnelles et de mise en œuvre des opérateurs élémentaires. Lorsqu'on combine des opérateurs élémentaire, il est assez rare de pouvoir mettre en commun des transistors. Il faut même en ajouter chaque fois que le couplage de deux opérateurs n'est pas possible directement en raison de l'utilisation de grandeurs physiques distinctes, ou de plages de variations différentes. Il en résulte une surface occupée qui est plus qu'additive par rapport à celle de ses constituants fonctionnels. La spécificité est également un obstacle à la réalisation de systèmes polyvalents : le jeu de commande des opérateurs se limite souvent à quelque(s) constante(s) d'espace ou de temps.

Choisir les bons opérateurs analogiques, les interfacer, en imaginer de nouveaux pour de meilleures performances en surface et en puissance consommée, tous ces aspects font que la conception analogique reste très délicate. Par rapport à la conception de circuits numériques, les différences marquantes sont la multiplication des aléas, la pauvreté des outils de CAO et une testabilité précaire. Cela aboutit souvent à plusieurs cycles de fabrication des circuits avant d'avoir un résultat opérationnel. Finalement, l'investissement n'est rentable que pour des structures d'une certaine universalité, que celle-ci soit de nature calculatoire ou visuelle. Seule la production de masse peut justifier des conceptions plus spécifiques.

### Les circuits analogiques programmables

Deux approches différentes coexistent parmi les circuits analogiques dits universels (polyvalents et programmables), les réseaux de neurones cellulaires et les microprocesseurs génériques :

**Les réseaux de neurones cellulaires** Les **CNN**[17] sont un modèle de machine de calcul massivement parallèle **SIMD**[18] analogique définie en espace discret à $N$ dimensions. Les CNN reposent sur une loi d'évolution générique (inspirée du domaine de l'automatique) et utilisant une image comme variable. D'après [CR93] :

[17] *Cellular Non-linear Networks* en anglais.

[18] Single Instruction Multiple Data en opposition avec Multiple Instruction Multiple Data, désigne un mode de fonctionnement des ordinateurs dotés de plusieurs unités de calcul fonctionnant en parallèle.

- Un CNN est un tableau d'éléments (cellules) à $N$ dimensions ;
- la grille de cellules peut être par exemple un tableau à topologie rectangulaire, triangulaire ou hexagonale, un tore à 2 ou 3 dimensions, ou une séquence en 3 dimensions de tableaux à 2 dimensions (couches) ;
- les cellules sont des processeurs à multiples entrées et simple sortie, toutes décrites par un ou quelques paramètres fonctionnels ;
- une cellule est caractérisée par une variable d'état interne, quelquefois pas directement observable en dehors de la cellule ;
- plusieurs réseaux de connections peuvent être présents, avec différentes tailles de voisinage ;
- les CNN constituent un système dynamique qui peut opérer à la fois en temps continu (CT-CNN) et en temps discret (DT-CNN) ;
- les données et les paramètres des CNN sont généralement des valeurs continues ;
- les CNN opèrent généralement avec plusieurs itérations (ce sont des réseaux récurrents).

Les informations sont échangées uniquement de manière locale. Ce fait constitue la principale différence entre les CNN et les autres modèles de réseaux de neurones. Bien sûr ; ces caractéristiques permettent aussi d'obtenir des calculs globaux. Les communications entre les unités non connectées directement (les cellules éloignées) sont obtenues par communication à travers les autres unités. On peut considérer le concept de CNN comme une évolution du concept d'automate cellulaire. De plus, il a été démontré que le concept de CNN est universel, c'est à dire qu'il est équivalent à une **machine de Turing**[19]. Une description mathématique formelle est donnée par la formule suivante :

$$\begin{cases} x(t+1) = g(x(t)) + I(t) + \sum_k (A(y_k(t), PA(j))) + \sum_k (B(u_k(t), PB(j))) \\ y(t) = f(x(t)) \end{cases}$$

où $x(t)$ est l'état interne de la cellule à l'instant $t$, $y(t)$ est la sortie, $u(t)$ est l'entrée et $I(t)$ une valeur locale appelée biais. $A$ et $B$ sont deux fonctions génériques, $PA(j)$ et $PB(j)$ sont des matrices de paramètres (typiquement le poids des connections entre les cellules). Les valeurs des voisins $y_k$ et $u_k$ sont collectées dans les deux directions de voisinages $N_r$, pour la fonction de rebouclage $A$, et $N_s$, pour la fonction de contrôle $B$. Les deux voisinages sont potentiellement différents. Les fonctions $A$ et $B$ sont aussi appelées patrons

[19]Une machine de Turing est un modèle abstrait du fonctionnement d'un ordinateur et de sa mémoire, créé par Alan Turing en vue de donner une définition précise au concept d'algorithme ou "procédure mécanique". Ce modèle est toujours largement utilisé en informatique théorique, en particulier pour résoudre les problèmes de complexité algorithmique et de calculabilité.

(" *template*" en anglais). La fonction de rebouclage local instantané $g$ exprime la possibilité d'un effet de rebouclage immédiat. Cette fonction n'est généralement pas utilisée. $f$ est la fonction qui fixe la sortie de la cellule d'après son état interne. Par convention on utilise comme fonction interne la distance " *chessboard*" exprimée par l'équation :

$$d(i,j) = max(|x(i) - x(j)|, |y(i) - y(j)|)$$

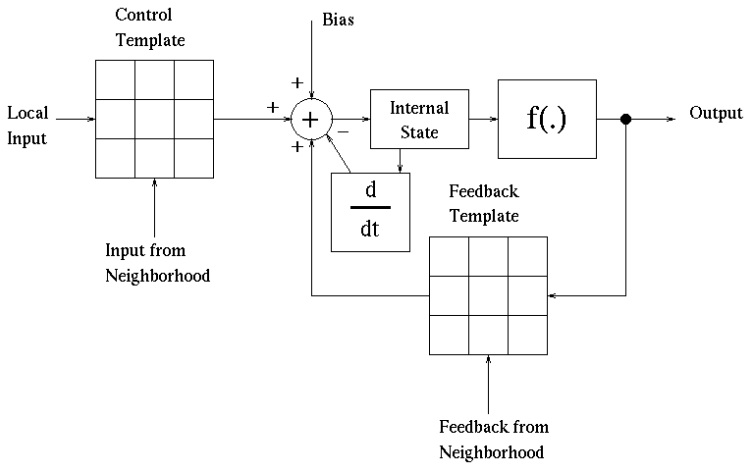



FIG. 1.3 – Le schéma bloc d'une itération d'un CNN générique.

la Figure 1.3 présente le schéma bloc d'une itération d'un CNN générique. Dans la plupart des cas le système est non-markovien, c'est-à-dire que les états internes futurs dépendent aussi des états passés du système. Pour les CNN variants dans le temps, toutes ces fonctions peuvent être aussi fonction du temps.

Une des limitations principales des réseaux de neurones cellulaires analogiques est le coût engendré par les fonctions destinées à programmer les coefficients des matrices de convolution. D'autre part pour effectuer des traitements complexes, il est nécessaire de séquentialiser les traitements effectués au sein des CNN. Cette migration nécessaire mais progressive vers le numérique s'est même étendue aux fonctions de traitement des CNN analogiques de certains projets. Toutefois, le mode de calcul par relaxation des CNN impose d'effectuer les calculs en parallèle sur tous les bits, ce qui conduit à utiliser des opérateurs numériques multi-bits dont le coût devient rapidement très important.

La limitation plus fondamentale des réseaux de neurones cellulaires réside dans sa propriété fondatrice. Ainsi, afin de permettre une implémentation du modèle, les interactions ne se font que de manière locale. De plus, pour effectuer des opérations régionales,

il est nécessaire d'effectuer une succession d'opérations locales itérées. Ces itérations sont une limitation du modèle dès lors que l'on s'intéresse aux opérations régionales. Des opérateurs permettant d'effectuer ces calculs régionaux sans itérations seraient préférables. Ils sont d'ailleurs présents de manière théorique dans d'autres modèles de réseaux de neurones. Toutefois, l'implémentation de ces opérateurs régionaux requiert un investissement matériel trop important, ce qui a conduit à son abandon dans le modèle des réseaux de neurones cellulaires. Cette limitation due au mode de communication local se retrouve dans les rétines numériques synchrones[20] à voisinage limité où les communications s'effectuent de proche en proche et dans les méthodes de traitement d'image par équations aux dérivées partielles, qui sont d'ailleurs très bien implémentées dans les réseaux de neurones cellulaires. Pour plus de détails sur les réseaux de neurones cellulaires, le lecteur pourra se reporter aux références suivantes : [MM88][BA92] [LFE+99].

**Les microprocesseurs génériques** Plus récemment deux projets de circuits basés sur une réalisation de processeurs analogiques génériques ont vu le jour avec des approches assez différentes : l'un cherche à intégrer un processeur comparable à un processeur numérique (Université de Manchester) ; l'autre préconise l'implantation de processeurs dotés à la fois d'une unité de calcul analogique et numérique (projet PARIS[21] de l'Institut d'Electronique Fondamentale de l'Université Paris XI Orsay.).

Le **microprocesseur générique analogique** (appelé A$\mu$P)[DH00] travaille sur des données analogiques échantillonnées et dispose d'un contrôle numérique. Le A$\mu$P exécute des programmes de la même manière que le ferait un microprocesseur numérique, mais en manipulant des données analogiques échantillonnées dans ses unités de calcul et sur ses chemins de données au lieu de données binaires codées sur plusieurs bits. Il est capable d'échanger d'une part des données analogiques entre ses registres (copies) en exécutant de manière séquentielle les instructions qui lui sont envoyées par le contrôleur externe à travers ses ports d'entrées / sorties, à destination de voisins. D'autre part, il est capable d'effectuer des calculs sur ces données, y compris des comparaisons et des branchements conditionnels.

[20] Le synchronisme désigne le caractère de ce qui se passe en même temps, à la même vitesse. L'adjectif synchrone définit deux processus qui se déroulent de manière synchronisée.
- Il est particulièrement important dans le domaine technique, celui de l'électrotechnique, l'informatique et l'électronique. Il s'utilise aussi dans le domaine de l'audiovisuel pour la connectique.
- Les systèmes informatiques sont fréquemment fondés sur ce principe, qui permet des transferts d'information fiables, que ce soit au niveau des réseaux ou au niveau du processeur (bien qu'il existe quelques processeurs asynchrones). Du point de vue du programmeur, une méthode est synchrone si elle attend une réponse avant de retourner la sienne.

[21] Programmable Analog Retina-like Image Sensor

La figure 1.4 présente l'architecture globale du microprocesseur sous la forme d'un diagramme fonctionnel. Elle ressemble fortement à celle d'un processeur numérique. Le processeur s'articule autour d'un bus analogique qui sert à interconnecter un banc de registres analogiques, une ALU[22] et une unité de communication (entrées et sorties analogiques). Chaque module de cet ensemble (ALU, registres et unité de communication) étant contrôlé par un ensemble de signaux numériques jouant le rôle d'instructions, envoyés par un contrôleur externe.

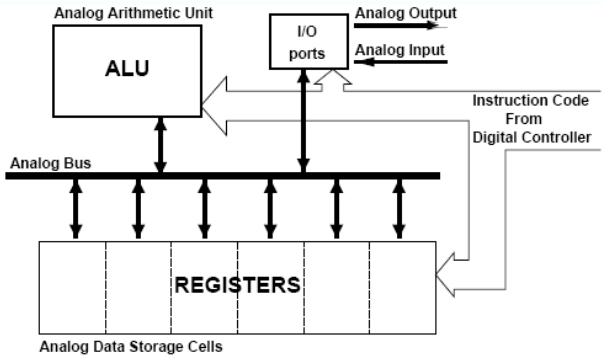



FIG. 1.4 – Schéma bloc du processeur analogique (A$\mu$P)

Le A$\mu$P est capable d'exécuter séquentiellement (avec une synchronisation externe de type numérique) des échanges de données analogiques avec les voisins et des calculs sur ces données analogiques.

L'ensemble forme ainsi un système mixte analogique / numérique capable d'opérer en mode SIMD. Le principe de fonctionnement du processeur A$\mu$P repose sur les techniques dites de communication de courants[23]. Ces solutions s'avèrent compactes et performantes, dans la mesure où la sommation et la soustraction de courants sont réalisées facilement sur un fil en un nœud, conformément à la loi de Kirschoff[24]. Le stockage des données dans les registres présente une perte de mémoire. En effet, lorsque des algorithmes récursifs sont exécutés, l'erreur ainsi engendrée à chaque transfert de données risque de mettre en péril la viabilité de l'algorithme. En terme de compacité, les microprocesseurs génériques sont de tailles comparables aux CNN. En terme de performances, les fréquences sont assez basses. Enfin en matière de consommation d'énergie, la nécessité d'utiliser des transistors dans leur zone de saturation rend impossible la réalisation de matrices de grande dimension.

[22]Unité Arithmétique et Logique, en fait AAU serait probablement plus approprié pour désigner cette unité de calcul arithmétique analogique (Arithmetic Analog Unit) même si elle permet de gérer des comparaisons pour l'exécution de code conditionnel.

[23]Switched-current, en anglais, noté en général SI.

[24]égalité du courant circulant dans une boucle

Le projet **PARIS** [KDMA99] de l'Institut d'Electronique Fondamentale à l'Université de Paris XI fait appel à des solutions différentes de celles exposées ci-dessus dans le cadre du processeur analogique générique sur données échantillonnées. L'objectif de ce projet consiste en la conception d'une nouvelle génération de rétine programmable ayant recours à un processeur doté d'une unité de calcul analogique, mais aussi d'une unité de calcul booléen. La complexité du processeur envisagé ne permet pas son intégration en chaque cellule de la matrice de capteurs, mais il est destiné à être intégré en bordure, sous la forme d'une ligne de processeurs SIMD. Cependant, l'originalité de l'approche proposée ne s'arrête pas là. La matrice de capteurs joue aussi un rôle de mémoire analogique : chaque cellule contient 4 registres de stockage pour les données analogiques, l'une d'entre elles servant à intégrer le courant en provenance du photorécepteur. Le photorécepteur est constitué de deux phototransistors verticaux de taille minimale associés en parallèle, afin de bénéficier d'une grande sensibilité et d'une grande bande passante, le courant décharge la capacité préchargée à une tension de référence. Le processeur, en bordure, est connecté aux bus mixtes analogiques/numériques de 3 colonnes successives, à travers un multiplexeur 3 vers 1, afin de pouvoir effectuer facilement des calculs sur un voisinage local. Il se compose d'une unité de calcul analogique (voir Figure 1.5) qui exécute des instructions sous le contrôle d'un registre de condition 1 bit, d'une unité logique et d'un banc de registres mémoire mixtes analogiques/numériques.

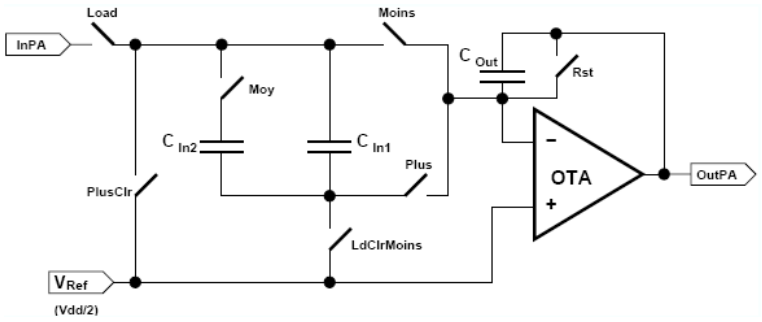



FIG. 1.5 – Schéma de principe de l'unité de calcul analogique dans le projet PARIS.

Les traitements accessibles sont très nombreux, en raison de la présence d'une unité de calcul et de moyens de stockage, tant numériques qu'analogiques. Les utilisations proposées sont les algorithmes de type CNN. La possibilité de conditionner les opérations analogiques par une combinaison booléenne permet d'envisager l'implantation de filtres anisotropes et d'opérations linéaires par morceaux, voire d'algorithmes stochastiques[25] en implantant un générateur aléatoire.

L'approche utilisée ici ne peut être considérée comme purement analogique, mais

[25]Le calcul stochastique est l'étude des phénomènes aléatoires dépendants du temps

résolument mixte. En terme de compacité, la dimension relativement grande choisie pour le pixel est surprenante étant donné l'absence de structure de traitement.

Deux raisons peuvent être avancées :

1. la taille des capacités de stockage pour la mémorisation analogique dans le pixel doit être importante, afin de conserver une précision acceptable,
2. une taille de pixel trop petite compliquerait fortement la conception du processeur en bordure à compter d'un par colonne, imposant une forme toute en longueur.

Dans les deux microprocesseurs génériques considérés, l'analyse des points faibles met en évidence les problèmes engendrés par les fonctionnalités analogiques. L'utilisation des transistors dans leur zone de saturation conduit à une surconsommation, et l'utilisation de capacités précises conduit à une vitesse faible et un encombrement élevé. Finalement, on peut se poser légitimement la question de la pertinence de l'utilisation de l'analogique dans les rétines artificielles. Celui-ci provient en effet principalement d'un héritage né avec l'analogie rétinienne biologique. Malgré cet héritage, l'intérêt du numérique s'est néanmoins peu à peu imposé dans le monde des rétines analogiques.

### Les circuits numériques polyvalents

L'Université de Stanford en Californie réalise dès 1999 des circuits de traitements de l'image purement numérique passant par l'intégration d'un convertisseur analogique / numérique dans le pixel. L'équipe utilise alors un convertisseur à modulateur $\Sigma$-$\Delta$ à un bit. Nous reviendrons par la suite sur un tel modulateur dans une application de détection du mouvement (Chapitre 3 section 3.6).

On peut citer comme circuits numériques polyvalents :

1. Le **CPA**[26][MA99], développé par Intel, est une rétine artificielle programmable au sens large puisque, comme son nom l'évoque, la nature du parallélisme mis en œuvre est linéaire et installé en bordure de la matrice de photodétecteurs. La particularité de ce circuit est qu'il n'embarque pas un processeur par colonne de pixels de la matrice de capteurs CMOS APS, mais un processeur 8 bits par bloc de 8 colonnes de photocapteurs. Contrairement à l'approche habituelle où un processeur est relié à chaque colonne, le pari est fait ici de réduire le parallélisme, afin de ne pas limiter les traitements à des calculs bit-série et d'exploiter la surface disponible pour implanter un processeur d'architecture conventionnelle et souple d'utilisation. D'autre part, afin d'être en mesure de réaliser un circuit de résolution élevée, le choix d'un pixel

[26] Column-based Processing Array.

de faible dimension est fait. Chaque colonne dispose d'un convertisseur 8 bits de type Σ-Δ, dont la conception semble avoir posé problème en raison de la faible largeur qui lui est imposée (la largeur du pixel)(Figure 1.6).

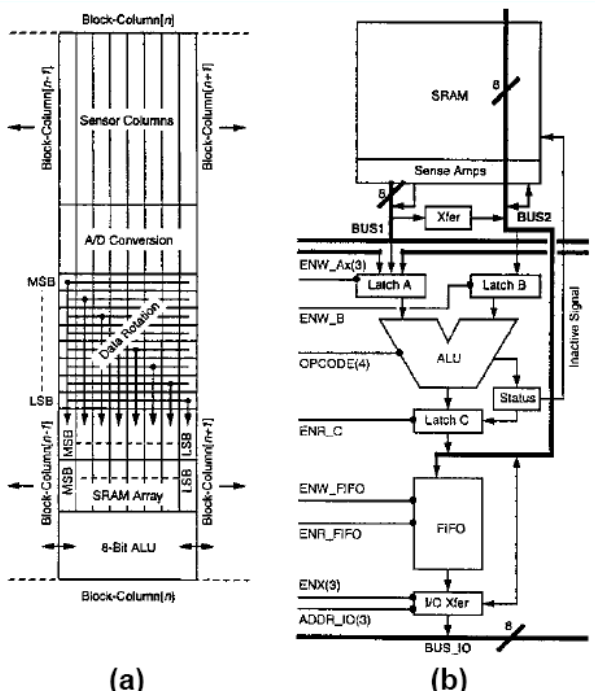



FIG. 1.6 – Architecture du circuit CPA d'Intel : (a) représente l'architecture d'un bloc de 8 colonnes avec ses 8 convertisseurs, sa mémoire de stockage (horizontale), après rotation des données et son processeur 8 bits ; (b) présente en détail l'architecture du processeur 8 bits, intégré pour chaque bloc de 8 colonnes.

On ne peut donc pas comparer directement ce type de circuit avec les autres circuits de type rétines numériques car le CPA ne dispose pas d'une architecture distribuée.

2. Le projet **SIMPil**[27][CGE+96] de l'Institut Technologique de Georgie (Georgia Tech, Atlanta, USA).

   Le système SIMPil intègre une matrice de processeurs numériques de type RISC très simples dans un circuit VLSI CMOS, sur lequel une matrice de capteurs photosensibles est hybridée (voir figure 1.7). Un ensemble de photocapteurs est relié à chaque noeud (microprocesseur) de la matrice. Le nombre de capteurs qu'il est souhaitable d'associer à chaque nœud a été étudié. Seul le prototype d'un nœud (microprocesseur, convertisseur Analogique / Numérique et électrode de contact) a été réalisé. L'étude des caractéristiques de diverses tailles de matrices et des répartitions en

[27]SIMD Pixel Processor.

nombre de capteurs par processeur a été réalisée à partir de simulations et d'interpolations. Le nœud de base est constitué du chemin de données du microprocesseur associé à un groupe d'échantillonneurs bloqueurs et un convertisseur A/N, utilisés pour échantillonner de manière synchrone l'ensemble des capteurs de la matrice et lire/encoder sur 8 bits le signal en provenance des capteurs qui lui sont associés.

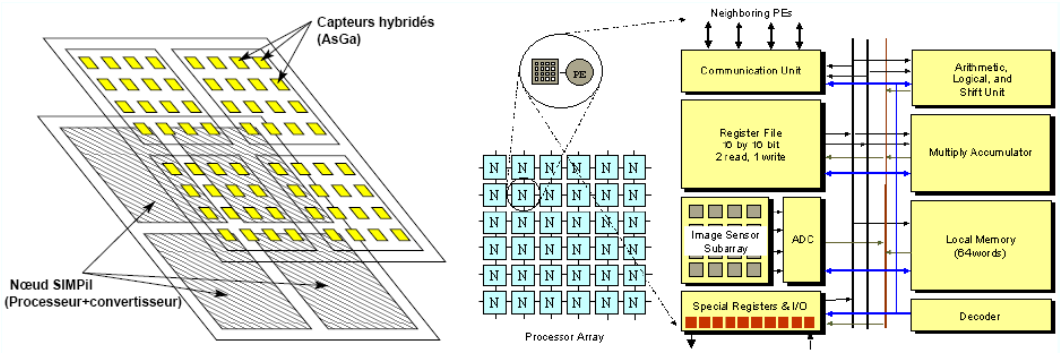



FIG. 1.7 – Architecture du SIMPil : à gauche, un circuit contenant 4 nœuds et la matrice de capteurs hybridés par dessus (16 capteurs sont associés à chaque nœud) ; à droite, une représentation en blocs fonctionnels d'un nœud.

Pour une complexité de traitement donnée et une taille d'image fixée, plus le nombre de capteurs associés à chaque nœud est grand, plus la puissance de calcul de chaque nœud doit être élevée et la fréquence de fonctionnement aussi. D'une manière générale, plus le nombre de capteurs associés est réduit (avec la borne inférieure égale à 1 : un processeur par pixel de l'image), plus l'énergie consommée est réduite (parallélisme accru et fréquence de fonctionnement nécessaire réduite). Cependant, étant donné la surface occupée par un nœud, la résolution accessible sur un seul circuit est très limitée et le recours à l'utilisation de plusieurs circuits est proposé.

On peut relever plusieurs contradictions entre les objectifs annoncés et les motivations réelles observées dans ce projet. L'objectif annoncé concerne la réalisation de systèmes de traitement d'image petits, légers et performants. Or l'étude réalisée ne porte que sur le développement du processeur du noeud, bien que la viabilité de l'assemblage d'un système complet y faisant appel, avec des résolutions acceptables ne soit pas démontrée de manière évidente. En effet, de grandes libertés sont envisagées quant aux aspects d'acquisition de l'image. L'utilisation de plusieurs circuits est proposée pour obtenir une image de résolution souhaitée avec les performances en calcul requises. D'une part, une telle approche entraîne des complications, en matière de calibration des différents circuits, ainsi qu'au niveau du développement d'une optique satisfaisante, sans évoquer les problèmes d'encombrement et fragilité d'un tel système. D'autre part, les seules limitations de consommation énergétiques qui soient considérées sont les maxima supportés par la technologie utilisée. Il est

fait abstraction de l'accroissement du bruit thermique dans les capteurs et du besoin de recourir à des moyens de refroidissement à la fois coûteux et encombrants. Ainsi, bien que l'utilisation dans des systèmes mobiles soit évoquée, aucune motivation précise ne semble apparaître pour mettre au point une solution viable tenant compte des contraintes que cela implique. La motivation semble être de nature purement technologique et aucune application précise n'est visée.

3. Le **système de vision milliseconde** de l'Université de Tokyo au Japon vise à l'implémentation de l'unité de traitement au niveau du pixel, soit un parallélisme comparable au circuit sur lequel nous effectuons nos travaux [NITM00][KIIY03] [KKI04]. L'objectif de ce projet est la réalisation d'un système de vision rapide pour des applications en robotique.

   La complexité de l'élément de calcul introduit dans chaque pixel est bien moindre en comparaison de tous ceux rencontrés jusqu'à présent dans cette section comme l'atteste la Figure 1.8. Cependant elle reste relativement inspirée par une structure conventionnelle de processeur, bien que fortement dépouillée. Le processeur intégré ici reçoit des instructions sur 5 bits qui sont décodées dans le pixel lui-même.

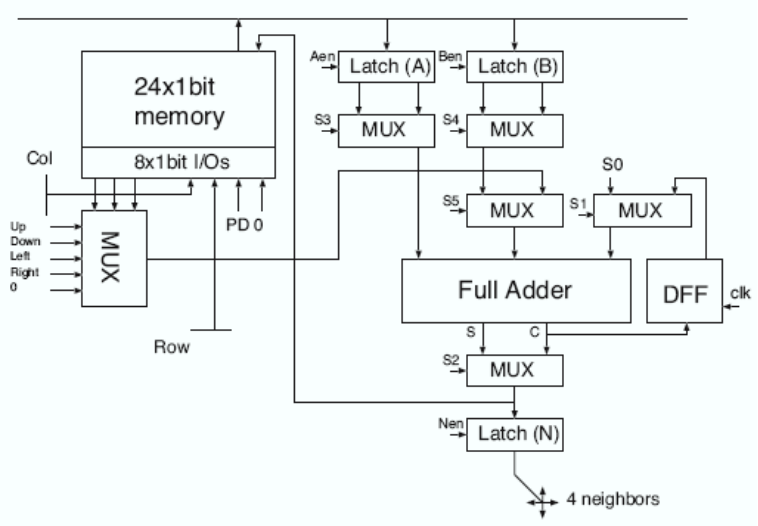



FIG. 1.8 – Structure du processeur élémentaire du circuit prototype 64x64 pixels 0.35$\mu$m de l'Université de Tokyo.

Ainsi le pixel se compose d'une photodiode[28], comme élément photosensible et du PE constitué d'un banc de mémoire SRAM de 24 bits et d'une ALU réalisée à partir d'un *Full Adder* complet auquel sont connectés deux verrous en entrée et

[28]Une photodiode est un composant semi-conducteur capable de détecter un rayonnement, qui capte le signal optique et le transforme en signal électrique

un verrou en sortie. Les communications sont gérées par *mapping* mémoire (des adresses spéciales sont affectées pour récupérer la donnée d'un processeur dans le plus proche voisinage Nord-Est-Ouest-Sud ou la lecture seuillée du capteur).

4. Les circuits **LAPP**[29] **MAPP**[30], et la rétine **NSIP**[31] de l'Université de Linköping en Suède.

   Le circuit LAPP1100, commercialisé à partir de 1987 se compose d'une ligne de 128 pixels contenant chacun un processeur commandé en mode SIMD. Le signal provenant du photocapteur (photodiode fonctionnant en intégration) peut être seuillé et stocké dans un des 14 registres binaires que comprend le processeur ou envoyé directement sur l'unité de traitement. Celle-ci se compose de trois unités distinctes par lesquelles chemine la donnée et au travers desquelles elle peut subir un traitement, le résultat étant stocké dans un accumulateur. Les trois unités sont de natures différentes, ainsi que la portée des traitements réalisés :
   - la GLU[32], met en oeuvre des propagations le long de la ligne de processeurs, éventuellement conditionnées par le contenu de l'accumulateur,
   - la NLU[33], réalise des traitements locaux impliquant les voisins de droite et de gauche sur la ligne, sous la forme de masques,
   - la PLU[34], qui est capable d'effectuer un ensemble de fonctions booléennes de deux variables (ET, OU, OU EXCLUSIF...). L'ensemble du circuit est contrôlé par un système externe, chargée du séquencement des instructions.

   Introduit en 1991, MAPP2200 reprend sensiblement la même architecture de traitement en lui associant non plus une ligne de pixels, mais une matrice de 256x256 pixels. La nature du parallélisme a donc changé, puisqu'un processeur seulement est assigné à une colonne entière de pixels au lieu d'un à chaque pixel. La ligne de processeur qui s'est vue agrandir à 256 processeurs obéit au même jeu d'instructions (compatibilité ascendante) et fait toujours appel aux unités GLU/NLU/PLU et à des calculs bit-série. Par contre le processeur s'agrémente de registres supplémentaires pour un total de 114 registres numériques (96 registres généraux, 16 registres spéciaux ainsi qu'un accumulateur et un registre de retenue) et un registre analogique pour la lecture du photocapteur. Cette augmentation du nombre de registres est la bienvenue pour combler la différence entre le nombre de processeurs et de pixels à traiter (chaque processeur doit gérer une colonne entière de pixels) ainsi que pour être en mesure de manipuler des images en niveaux de gris. Un convertisseur A/N simple à rampe a été choisi pour encoder en binaire naturel, sur 2 à 8 bits, les niveaux de gris de l'image (ensuite manipulés sous la forme de calculs bit-série). Enfin un

[29] Linear Array Picture Processing.
[30] Matrix Array Picture Processing.
[31] Near Sensor Image Processing.
[32] Global Logical Unit.
[33] Neighbor Logical Unit.
[34] Point Logical Unit.

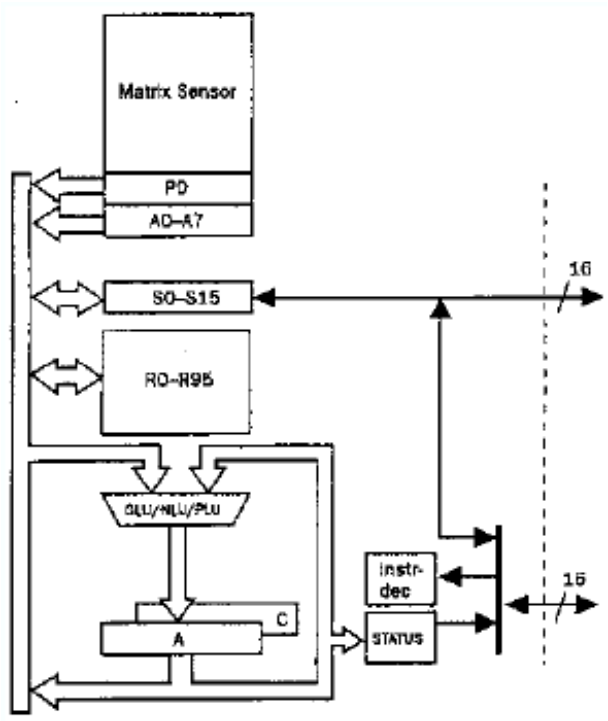



FIG. 1.9 – Architecture générale du MAPP2500.

ensemble de 8 registres à décalages (spéciaux) a été ajouté pour déplacer les données images (sur 8 bits) en interne, mais surtout procurer un moyen supplémentaire d'extraction de l'image vers l'extérieur, contrôlé de manière autonome (permettant un recouvrement entre l'extraction de données et les traitements).

Introduit en 1999, MAPP2500 étend la taille de la matrice de photocapteurs du circuit MAPP2200 à 512x512 pixels et la ligne de processeurs à 512 processeurs. Aucun changement dans l'architecture (Figure 1.9) n'est observé excepté la transformation de 8 registres spéciaux qui se joignent aux 8 registres à décalage précédents, pour former 16 registres à décalage, utilisés pour accroître les performances des entrées/sorties (ceci est rendu nécessaire par l'augmentation de la taille de la matrice).

Les circuits MAPP (Figure 1.9) jouissent d'une bonne réputation dans le domaine du contrôle industriel, domaine pour lequel ils sont plus adaptés que celui de la vision en général, en raison d'une dépendance particulière vis à vis de l'orientation des traitements, qui découlent du parallélisme de ligne (ou colonne) mis en œuvre. Mais les traitements spatio-temporels réalisables sont limités, en raison de l'impossibilité de stocker ne serait-ce qu'une image complète, nécessaire pour faire interagir des données images espacées dans le temps. L'équipe avait écarté la possibilité d'implanter un processeur capable de traiter des images en niveaux de gris, dans chaque pixel, en raison de la trop grande quantité de ressources nécessaires estimée jusqu'alors. Le concept du NSIP[35][ESA96] l'amène à réévaluer sa position et la conduit à la réalisation en 1996 d'une rétine NSIP. Ce concept permet de traiter des images en niveaux de gris en ne manipulant que des données binaires en sortie du capteur, et cela, sans nécessiter le stockage systématique du niveau de gris sous forme explicite (binaire naturel sur 2 à 8 bits par exemple). Ce concept vise à encoder les niveaux de gris par le temps, grâce à la capacité d'interroger de manière successive l'état du capteur (sans le perturber). Ce sont les coupes de l'ombre de l'image qui sont alors obtenues et peuvent être utilisées pour effectuer des traitements en niveaux de gris (c'est le même formalisme qui a permis d'étendre le domaine des traitements de la morphologie mathématique, initialement binaire, aux niveaux de gris). Nous présentons plus en détails ce type de calcul dans la section 2.2.1.

### Les rétines TCL

Les travaux de recherches sur l'algorithmique **TCL**[36] commencés au début des années 80 au l'Etablissement Technique Central de l'Armement (ETCA) d'Arcueil[Zav81] et à

[35]Near Sensor Image processing.
[36]Traitements Combinatoires Locaux.

l'IEF[Gar84][DGZ85] aboutissent en 1984 à la réalisation d'un circuit appelé la rétine TCL de première génération. La classe des traitements TCL découle logiquement d'une observation des caractéristiques d'invariance par translation et de localité des traitements d'image de bas niveau. Le processus d'acquisition est homogène et se déroule simultanément et de manière identique sur chaque pixel. Les traitements réalisés sur les pixels acquis sont indépendants de leur position dans l'image (invariance par translation). L'information dans l'image conserve une certaine localité ce qui fait que l'interaction entre deux pixels a d'autant moins de sens que ceux-ci sont éloignés dans l'espace et/ou dans le temps, d'où la limitation des traitements à de petits voisinages (locaux). Un TCL correspond au calcul d'une fonction booléenne invariante par translation sur une image binaire, dont l'application à chaque pixel entraîne le remplacement de la valeur du pixel par une combinaison logique de sa valeur courante et de celles de certains de ses voisins. Disposer d'une architecture massivement parallèle SIMD avec un élément de calcul par pixel, permet l'exécution d'un TCL simultanément sur tous les pixels d'une image.

Le calcul d'une fonction booléenne telle qu'un TCL nécessite peu d'opérateurs pour s'implanter. Sous *forme normale disjonctive* (FND), toute fonction booléenne s'exprime en utilisant au plus 3 bits de mémorisation et exige simplement la disponibilité d'un opérateur de complémentation, d'un opérateur de conjonction pour calculer le monôme en série et d'un opérateur de disjonction, afin d'accumuler tous les monômes. La Figure 1.10 montre comment une FND est calculée sur la rétine : le plan du haut contient l'image binaire de départ. Pour chaque monôme, on lui fait décrire par des translations l'ensemble des variables qui constituent le monôme, en inversant l'image chaque fois que la variable correspondante apparaît complémentée. On calcule le "ET" logique de l'ensemble des variables de chaque monôme sur le plan du milieu, puis on calcule le "OU" logique de l'ensemble des monômes sur le plan du bas.

Pour implanter un TCL dans une architecture SIMD où une unité de calcul est présente dans chaque pixel, il est nécessaire, afin d'aller chercher les différentes variables, de faire subir un décalage à l'image binaire de départ. Ceci est facilement réalisable en utilisant une structure de registres à décalage semi-statiques. L'image peut ainsi être stockée et déplacée.

Poursuivant les objectifs du projet de rétine TCL, le centre Technique d'Arcueil de la DGA[37] met ensuite au point une deuxième génération de ces rétines. Partant du constat que le nombre de signaux d'horloge contrôlant le processeur TCL de la première génération limite sa densité d'intégration, les développements s'orientent alors vers une uniformisation du rôle des bits de mémorisation et la mise en commun d'un seul et même opérateur de calcul. La conception de ce circuit est le résultat des travaux de thèse de T. Bernard

[37]Délégation Générale pour l'Armement.

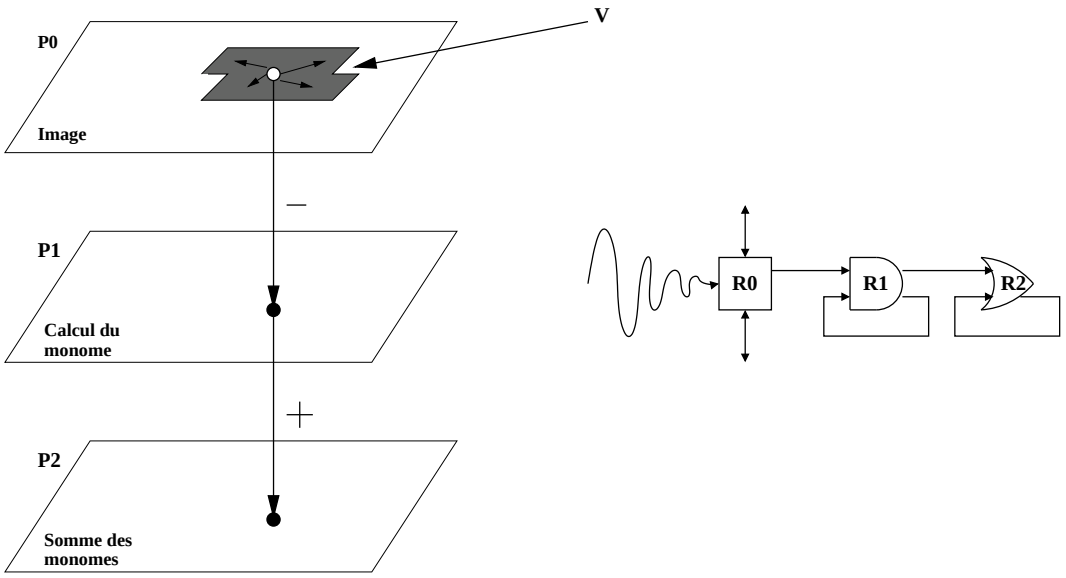



FIG. 1.10 – Calcul d'une Forme Normale Disjonctive sur 3 plans binaires.

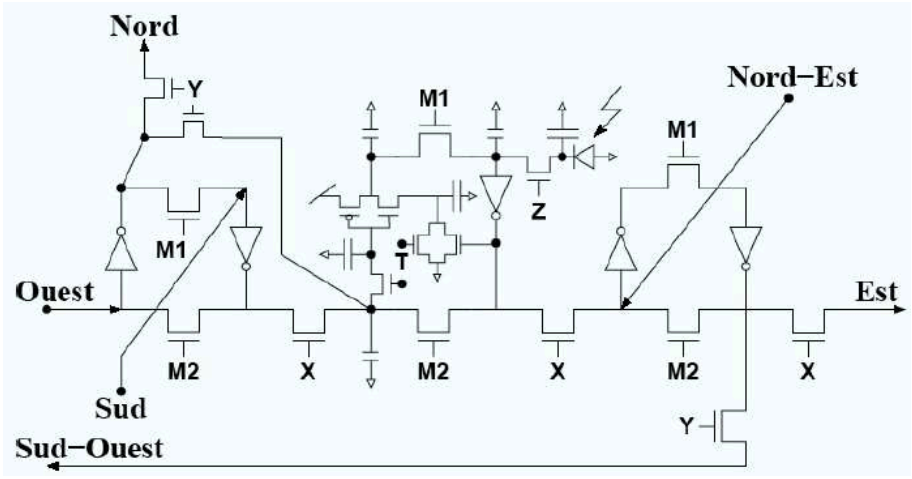



FIG. 1.11 – Schéma électronique du processeur élémentaire de la rétine TCL de deuxième génération [Pai01].

[Ber92], sa mise en œuvre, son test et son exploitation font l'objet des thèses de nature système dans les années suivantes[Ngu96].

Les efforts de minimisation du nombre de signaux nécessaires au contrôle du processeur TCL booléen ont porté leur fruits. Les 3 bits de mémoire dont dispose celui-ci font maintenant partie d'un seul et unique registre à décalage semi-statique bidimensionnel et tridirectionnel.

Cependant, il en résulte une grande difficulté de programmation et une inefficacité tant énergétique qu'algorithmique, en raison du grand nombre de déplacements élémentaires à réaliser pour obtenir un décalage quelconque, et du fait que tous les bits de mémorisation de la matrice sont systématiquement déplacés. Un programme fournissant une aide précieuse au programmeur a été développé afin d'optimiser les déplacements de données, calculant la plus courte séquence d'horloges nécessaire à l'obtention d'un déplacement particulier.

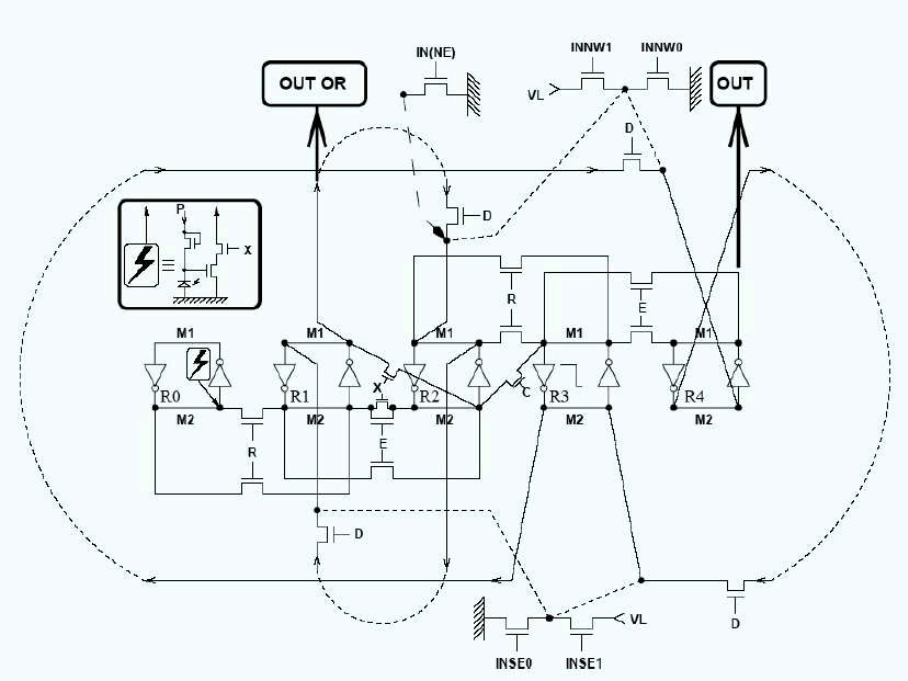



FIG. 1.12 – Schéma électronique du processeur élémentaire de la rétine PVLSAR [Pai01].

La période 1993-1997 est dédiée à la mise en œuvre et à l'exploitation algorithmique de cette rétine en s'intéressant aux aspects système dans la conception de machines de vision à base de **RANP**[38]. F. Paillet [Pai01] contribue par ses travaux de recherches à

[38]Rétines Artificielles Numériques Programmables.

la conception et la mise en œuvre d'une rétine de dimension plus importante appelée **PVLSAR**[39]. Cette rétine est constituée d'une matrice de 128x128 pixels, chacun ayant 5 bits de mémoire. La structure du processeur élémentaire (Figure 1.12) reprend celle de la rétine TCL de deuxième génération (Figure 1.11) avec certaines améliorations visant notamment à permettre la multigranularité des traitements et permettant de manipuler les données plus efficacement et pour un coût plus réduit.

Les rétines TCL de deuxième génération puis PVLSAR ont permit d'avoir un banc de test très significatif pour l'élaboration d'algorithmes de traitement d'image sur les rétines artificielles numériques programmables. La Figure 1.15 présente le banc algorithmique actuel ainsi que la dernière puce Pvlsar34 employée pendant nos travaux de recherche.

### Conclusion

Pour une revue plus complète des travaux réalisés dans le domaine des rétines artificielles, le lecteur peut se reporter aux travaux suivants [Moi97b] [Ber92] [PMB99] [Man00] [Elo05]. Si de nombreux projets entretiennent l'idée de traiter l'information au plus près du capteur afin de reproduire les mécanismes primaires de la vision animale, les difficultés techniques inhérentes à la réalisation d'un tel système découragent de nombreuses équipes et les amènent à intervenir à un plus haut niveau dans la chaîne du traitement de l'image. Les avancées technologiques en ce qui concerne l'intégration submicronique rendent aujourd'hui possible l'élaboration d'un circuit combinant capture et traitement de l'image. Dès lors, pourquoi ce principe ne se généralise t-il pas ? Les réponses sont multiples, mais l'argument principal est sans doute que leur coût de fabrication est encore élevé et que les compétences à mettre en œuvre pour maîtriser leur conception et leur utilisation restent difficiles à acquérir. Nous allons cependant montrer dans ce mémoire qu'il est possible à moindre frais de développer des algorithmes pour la rétine de façon efficace et ne nécessitant pas une formation spécifique.

Le schéma de la Figure 1.13 replace les différents projets de rétines artificielles les uns par rapport aux autres en fonction de leur architecture (analogique ou numérique) et de leur spécificité (dédié ou polyvalent). La surface des cercles représente l'activité que les différents projets ont suscités ses dernières quarante années.

Actuellement, plusieurs projets de réalisation continuent à être actifs, notamment au Japon (université de Tokyo), aux Etats-Unis (CalTech), et en Espagne (université de Séville). D'autre part, divers projets de circuits analogiques sont toujours étudiés en

[39] Programmable and Versatile Large Size Artificial Retina.

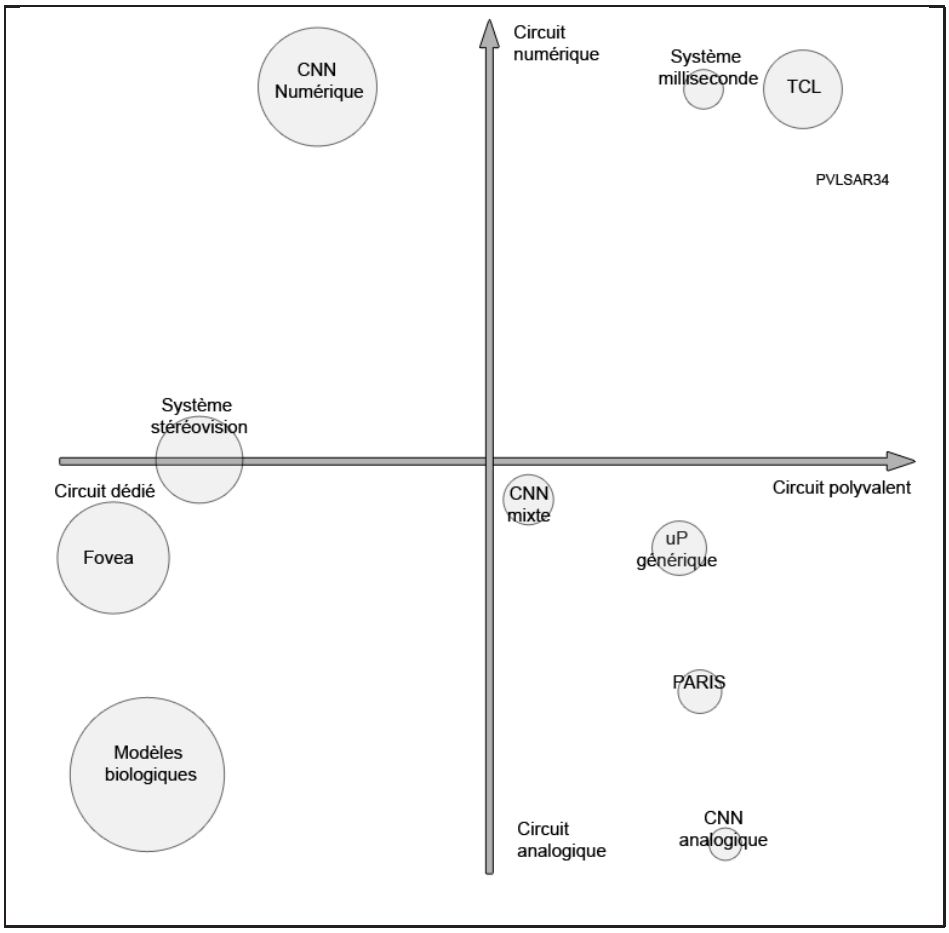



FIG. 1.13 – Résumé de l'état de l'art sur les rétines artificielles.
Les différents projets sont classés en fonction de leur architecture (analogique ou numérique) et de leur spécificité (dédié ou polyvalent). La taille des cercles représente l'activité que les différents projets ont suscitée.

association avec un FPGA[40] (à l'INPG notamment) mais avec une philosophie assez différente des circuits purement analogiques. Le schéma de la Figure 1.14 présente les projets de rétines numériques artificielles programmables actifs aujourd'hui. Sur l'axe horizontal est reportée la résolution (le nombre de pixels) du circuit et sur l'axe vertical la capacité mémoire du processeur élémentaire. La surface (grosseur) du point représente le rapport surface / complexité. Un petit point est donc préférable à un gros.

En fait l'idéal est d'avoir le plus petit point (donc la plus grande compacité) le plus à droite possible (donc la plus grande résolution) et le plus haut possible (donc la plus grande capacité mémoire). PVLSAR34 se comporte très bien et dispose de la plus grande résolution. De plus grâce à sa capacité de multigranularité et de multirésolution[41], il est capable de s'élever dans le graphique, en accroissant virtuellement sa capacité memoire. Nous discuterons de cette capacité dans la suite du mémoire.

### 1.2.3 La rétine artificielle numérique programmable Pvlsar34

Après ce rappel historique, il apparaît que la solution optimale consiste à choisir un système mixte analogique-numérique. En effet, la partie numérique induit une grande programmabilité et la partie analogique permet de paramétrer un grand nombre de fonctions, soulageant ainsi la "couche" numérique .

Cependant, un mélange aussi intime de fonctions analogiques et numériques nécessite une mise au point technologique très délicate et les rétines utilisées pour ce projet sont de ce fait essentiellement numériques. En particulier, les opérations élémentaires effectuées sur les images sont exclusivement booléennes.

Le modèle de rétine que nous avons étudié et avec lequel nous avons implanté nos algorithmes est la rétine PVLSAR 34 (Figure 1.15), dernière née de la famille des rétines PVLSAR, développée à l'ENSTA par Thierry Bernard en 2005. Il s'agit d'une machine massivement parallèle de 40 000 processeurs interconnectés selon une grille 200x200 en topologie 4-connexe. Chaque pixel, gravé en technologie 0.35$\mu$m, a une taille de 37.5$\mu$m de coté et une capacité mémoire de 45 bits. Cette rétine constitue aujourd'hui la plus grande de la famille PVLSAR jamais réalisée.

Ainsi, la rétine artificielle que nous utilisons est constituée d'une grille bidimensionnelle de "cellules", chaque cellule étant formée de l'association d'un élément photosensible

[40] *Field-Programmable Gate Array*, un circuit intégré qui peut être reprogrammé après sa fabrication.
[41] Ceci n'est cependant vrai que pour Pvlsar 2.2 mais pas pour Pvlsar 3.4.

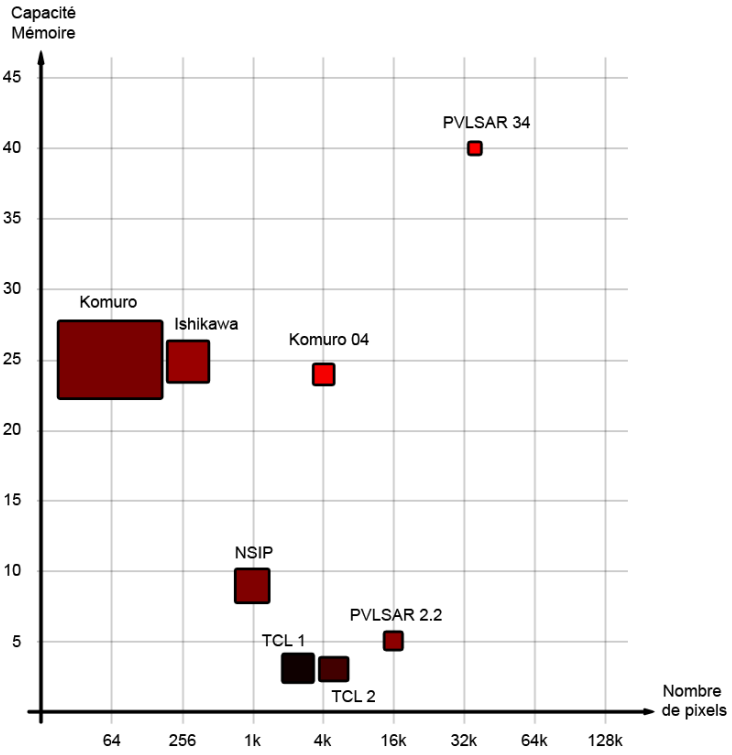



FIG. 1.14 – Comparaison des différents circuits RANP fabriqués à ce jour.
Les circuits sont classés en fonction de la taille de la matrice imageuse (en abcisse) et de la mémoire disponible dans le pixel (en ordonnée). La grosseur des points représente le rapport surface / complexité.

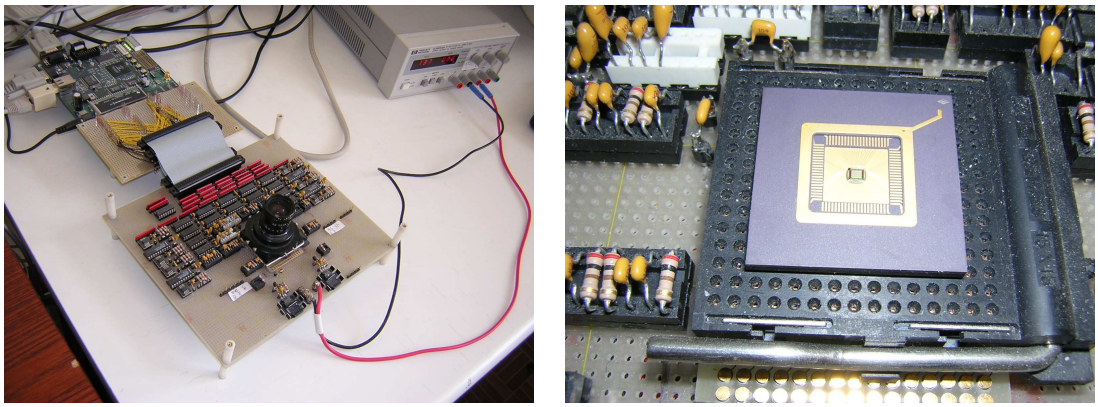


FIG. 1.15 – Une photo du banc algorithmique et de la rétine Pulsar32

(photorécepteur[42]), et d'un élément de calcul (processeur élémentaire (PE)). Chaque photocépteur est capable d'intégrer la lumière reçue (le nombre de photons) par une surface photosensible et de la transformer en courant électrique. Cette étape se nomme la **phototransduction**[43]. Ce courant est codé dans un certain nombre de registres binaires pendant la conversion analogique-numérique. Le signal binaire bidimensionnel ainsi obtenu est traité grâce aux opérations logiques et aux registres-mémoire au sein du processeur.

Le mode de parallélisme est purement SIMD, toute la grille étant commandée par un séquenceur externe à la rétine qui envoie une séquence d'instructions à la rétine selon une fréquence fixe, et à chaque pas de calcul, tous les 40 000 processeurs exécutent exactement la même instruction. Chaque cellule de la rétine exécute à chaque cycle d'horloge la même instruction élémentaire : soit une opération booléenne entre deux de ses registres mémoire (OU et inversion) ; soit une translation élémentaire vers un de ses quatre voisins dans les directions cardinales N, E, S, W. Chaque cellule d'une même ligne est rattachée à une même ligne de bus convergeant vers un palpeur. La mémoire est disposée sur un peigne formé par les lignes du bus.

Un programme rétinien est donc entièrement défini par la séquence d'instructions envoyée par le séquenceur. Chaque processeur est doté d'une mémoire numérique dont la vocation est de représenter les données d'un pixel. L'entrée des données se fait de manière optique : chaque processeur est pourvu d'un photocapteur et d'un mécanisme de conversion analogique - numérique qui lui permet de coder dans sa mémoire une grandeur numérique représentant une intensité lumineuse. Chaque processeur est aussi muni d'une unité de calcul permettant de lire, écrire et combiner de manière logique des données numériques à partir de et en direction de sa mémoire. Chaque couple de processeurs adjacents au sens de la 4-connexité partage une partie de leur mémoire, ce qui permet de communiquer des données entre pixels voisins.

Dans un système de vision, la rétine est associée à un hôte appelé cortex, typiquement composé d'un processeur scalaire (microprocesseur basse puissance DSP) et/ou d'un circuit de logique programmable (FPGA). Le cortex a deux fonctions :

- contrôler la rétine, en lui envoyant les séquences de commandes ;
- effectuer les tâches de haut niveau du problème de vision à partir des informations extraites de la rétine.

D'un point de vue technique, le cortex de la rétine actuelle est une carte de développement *Altera Excalibur EPXA1*[44]. Le dispositif EPXA1 contient un microprocesseur **RISC** (ou

[42]Une photodiode, diode à semi-conducteur dans laquelle un rayonnement lumineux incident détermine une variation de courant électrique

[43]la transformation physique du flux photonique en une grandeur physiquement exploitable.

[44]`http://www.altera.com/products/devkits/altera/kit-epxa1.html`

"*reduced instruction-set computer*) 32 bits **ARM922T**[45] combiné avec un FPGA APEX 20KE[46] dans un paquet FineLine BGA de 484-pin[47]. La Figure 1.16 présente la carte de développement Altera Excalibur EPXA1. Une mémoire *SRAM*[48] de 16Ko est partagée entre le processeur ARM et le FPGA. Dans la suite du document on appelera **nios**, l'ensemble formé par le processeur ARM et des composants électroniques permettant les entrées sorties entre la carte et l'extérieur (port série, LAN ethernet, ...).

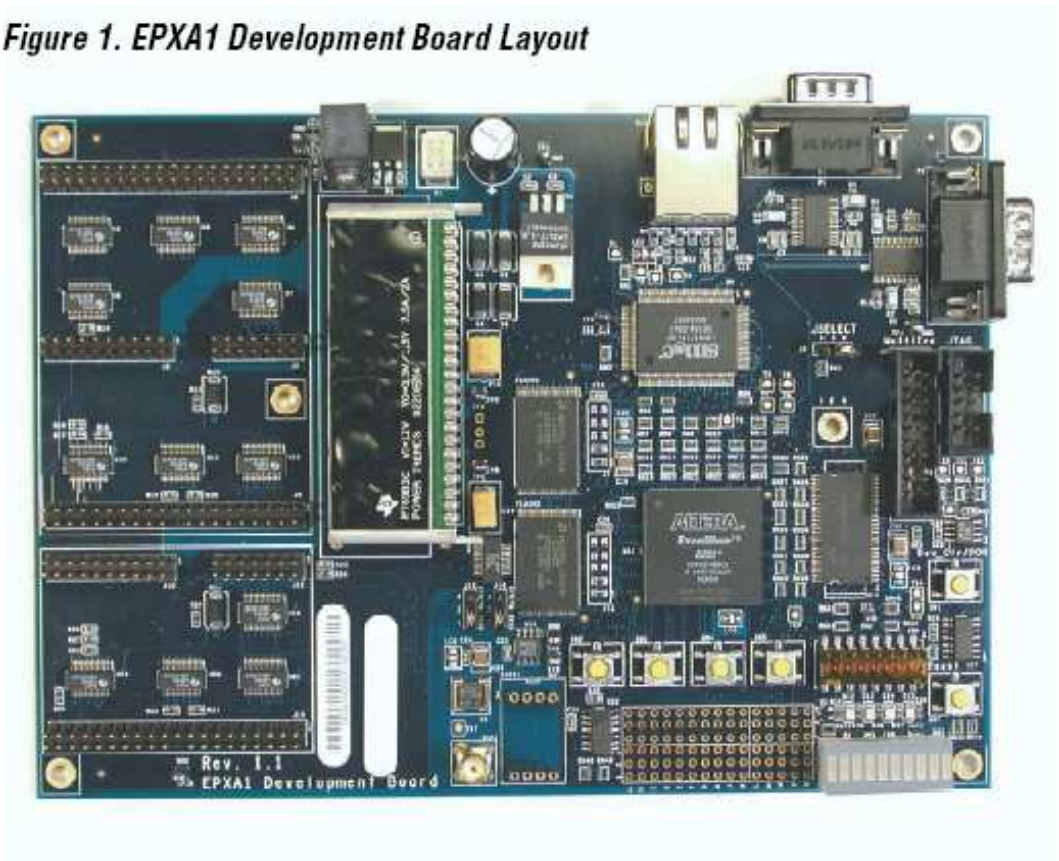



FIG. 1.16 – La carte de développement Altera Excalibur EPXA1.

Du point de vue des accès à la mémoire numérique de la rétine, il est important de comprendre qu'il n'existe pas de moyen de lire ou d'écrire depuis l'extérieur directement

[45]L'architecture ARM était initialement destinée à un ordinateur de la société *Acorn*, puis elle a été complétée pour devenir une offre indépendante pour le marché de l'électronique embarquée.

[46]`http://www.altera.com/products/devices/apex/overview/apx-overview.html`

[47]En micro-électronique, une matrice de billes (*ball grid array* ou BGA) désigne un type d'interconnexion entre le boitier d'un composant et le circuit imprimé

[48]La SRAM ou '*Static Random Access Memory*" est un type de mémoire vive dont les temps d'accès avait représenté, en leur temps, une avancée importante pour la rapidité des processus informatiques.

sur un pixel ou sur une zone spécifiée de la mémoire de la rétine. Le seul moyen d'accéder à la mémoire numérique de la rétine en lecture ou en écriture est par le bord de la grille, où le contenu de la mémoire d'une partie des processeurs est accessible par le bus de communication avec le processeur hôte (cortex). Par la suite, l'accès à un pixel ou une zone quelconque de la mémoire numérique se fait par combinaison de décalage effectués grâce au partage de la mémoire entre processeurs adjacents.

Chaque processeur étant associé à un élément photosensible du capteur, la surface qu'il occupe sur le circuit est très réduite. Par conséquent, la mémoire dédiée à chaque processeur est très faible et son jeu d'instructions très réduit. Dans la suite du document, nous appellerons **code rétinien** le code booléen exécuté sur la rétine. Il s'agit en fait d'un pseudo-code qui nous permet de tester et de valider les algorithmes que nous devons ensuite réécrire en code rétine véritable. Celui-ci se présente en suite d'instructions en assembleur spécifiquement développé pour le circuit.

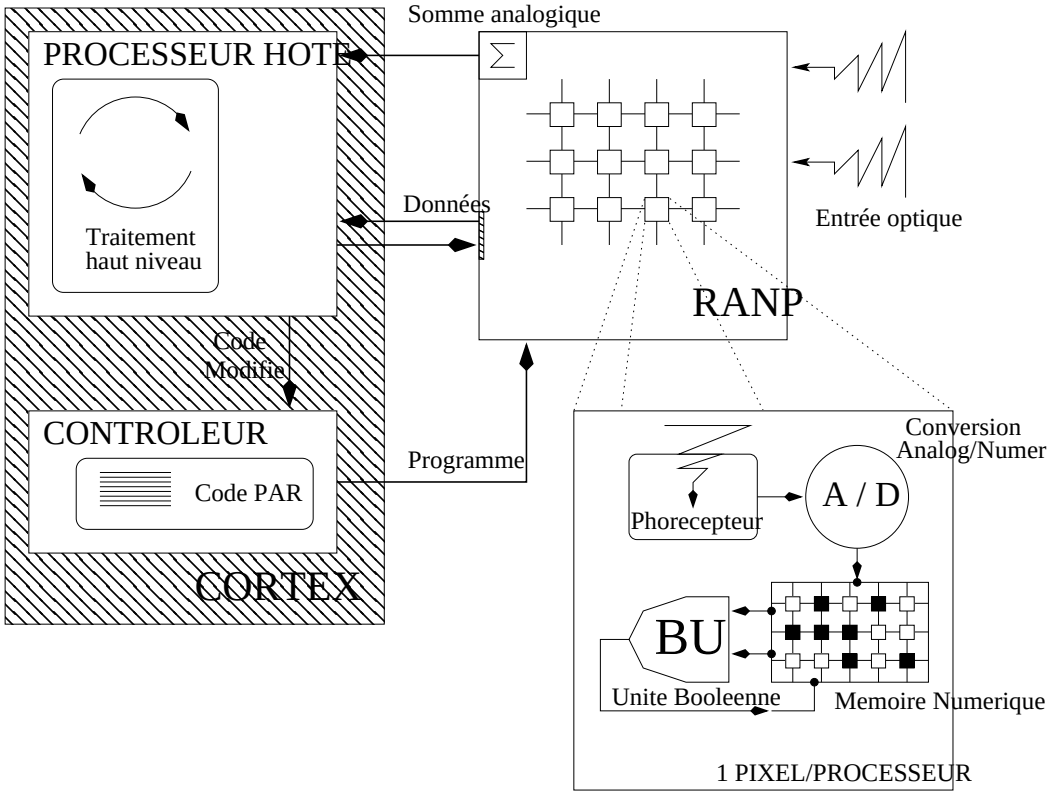



FIG. 1.17 – Schéma général de l'architecture du système composé de la rétine artificielle programmable (RANP) et du cortex avec un agrandissement du schéma d'un processeur élémentaire.

La Figure 1.17 fournit un schéma général de l'architecture du système composé de la rétine artificielle programmable et du cortex. Cette figure permet d'avoir un aperçu global du fonctionnement de la rétine et du rôle du dialogue rétine-cortex dans le déroulement des algorithmes que nous présentons dans la suite du document. Le processeur hôte charge le code compilé depuis l'extérieur puis le modifie afin d'envoyer des instructions parallèles au contrôleur qui se charge de séquencer le programme. La rétine exécute les instructions en effectuant des échanges de données avec le cortex. Concernant l'extraction de données, il y a deux manières d'extraire l'information de la rétine artificielle programmable :

1. par translation de l'image et par lecture la sortie sur le bord de la grille, afin d'obtenir la valeur d'un ou plusieurs plans de bits de la mémoire numérique.

2. par l'additionneur analogique global, qui fournit en temps constant une mesure approximative du nombre de "1" dans n'importe quel plan de bits de la mémoire numérique.

## 1.3 Conclusion

Dans ce chapitre nous avons présenté les différentes solutions de capteurs intelligents proposés ces quarante dernières années pour finalement présenter le circuit qui se trouve au centre de nos recherches, PVLSAR34. Il est à noter que la plupart des circuits présentés, a une ou deux exceptions prêts comme le Trackball Marble de Logitech, sont restés au stade expérimental et ont eu peu ou pas de débouchés industrielles. La diminution du coût de production de ces circuits ainsi que l'efficacité grandissant de ceux-ci pourraient cependant en faire des acteurs importants de l'industrie électronique et informatique dans les années à venir.

Nous rappelons que les principaux avantages offerts par un système de vision à base de rétine numérique artificielle programmable sont la **compacité du système**, sa **faible consommation** lui conférant une longue autonomie et enfin sa **vitesse de traitement** élevée due à l'intégration des éléments de calcul au sein du pixel et à son architecture massivement parallèle.

Après avoir introduit les aspects techniques de la rétine artificielle, le chapitre suivant centre notre propos sur son algorithmique particulière afin de poser définitivement les bases théoriques et techniques nécessaires à l'introduction du sujet réel de nos recherches : le **mouvement**. Nous présentons en particulier les bases arithmétiques du calcul rétinien et les outils adaptés à des algorithmes de plus haut niveau comme par exemple la morphologie mathématique.

# Chapitre 2

# Traitement d'images sur rétine numérique

## 2.1 Introduction

Avant de présenter les algorithmes de détection et d'estimation du mouvement qui se trouve au centre de notre travail, il convient d'introduire les traitements d'images binaires puis en niveaux de gris tels qu'ils sont implantés sur le circuit de la rétine numérique. Nous rappelons aussi brièvement les principes fondamentaux de la morphologie mathématique, ces outils étant abondamment utilisés dans les chapitres suivants.

En tant qu'architecture parallèle SIMD, la rétine présente une caractéristique dont nous avons déjà parlé et qui est très limitative : la taille réduite de la mémoire disponible au sein de chaque processeur. Sur une architecture classique, les opérations de base, celles qui sont implantées matériellement, sont des opérations arithmétiques alors que sur la rétine, les seules opérations de base sont des opérateurs booléens. Par conséquent, le formalisme décrivant les algorithmes pour la rétine programmable est celui des **treillis booléens**[1].

Ce principe fonctionnel de **granularité**[2] minimale constitue en fait l'originalité ma-

[1] Une algèbre de Boole se définit soit comme un treillis (un ensemble ordonné) distributif complémenté ; soit comme une structure algébrique à deux opérateurs possèdant les propriétés d'associativité, de commutativité, d'absorption et d'idempotence. Si l'on considère la première option, les propriétés d'associativité, ... n'ont pas de sens car ce sont des propriétés des opérateurs binaires et non de la relation d'ordre.

[2] La notion de granularité définit la taille du plus petit élément, de la plus grande finesse d'un système.

jeure de l'algorithmique que nous présentons. Il implique des difficultés pour repenser et adapter les algorithmes existants dans le but de l'implantation sur le circuit rétinien. Il permet aussi une **polyvalence** maximale, principe qui a motivé l'intégration de rétines programmables. Cette démarche est similaire avec celle employée avec le mode SIMD C* sur Connection Machine (POMPC)[Par93].

Notre travail se place dans un cadre de travail en quatre grande étapes :

1. utilisation de la rétine élaboré à l'ETCA et améliorée au LEI de l'ENSTA sous l'impulsion de Thierry Bernard[Ber92][Dul96][Ngu96][Pai01] ;
2. apprentissage et amélioration du code actuel de la rétine définie notamment dans les travaux de Antoine Manzanera[Duc99][Man00] ;
3. une étape de réflexion algorithmique afin de décider quelles sont les fonctionnalités que le circiut rétinien pourrait recevoir ;
4. et enfin, il nous reste la programmation sur la rétine des algorithmes conçue en adéquation avec les particularités de ce circuit.

On inistera dans ce chapitre sur le formalisme nécessaire à l'accomplissement de ces quatres étapes. Les critères d'evaluations utilisés afin dévaluer les performances de l'algorithmie r/'etinienne sont le temps de calcul et l'utilisation de la mémoire.

## 2.2 Principes de programmation

Dans le chapitre précèdent, nous avons vu que la rétine numérique Pvlsar 34 dispose de 48 bits (organisés en un tableau de 6 colonnes d'un octet (voir Figure 2.1)) de mémoire par pixel dont 4 bits sont en fait partagés par ses 4 pixels voisins. Le jeu d'instructions de la rétine est limité au calcul du "OU" logique entre 2 bits et à la négation d'un bit (Table 2.1). Afin de déplacer la valeur d'un registre du pixel $P(x, y)$ vers le pixel $P(x + i, y + j), (i, j) \in \{-1, 1\}^2, |i| + |j| = 1$, il suffit de placer la valeur du dit registre dans la case correspondant au déplacement puis de lire cette valeur dans la direction antagoniste sans oublier que tous les pixels effectueront cette opération.

Tous les registres mémoire sont accessibles en écriture et en lecture, cependant certains font l'objet d'une utilisation particulière. C'est le cas des points mémoire partagés entre deux pixels voisins et permettant la communication et les deux points mémoires qui, pour des raisons électroniques ne permettent de ne supporter que des demi-charges.

Quand on arrive au niveau de granularité d'un système, on ne peut plus découper l'information

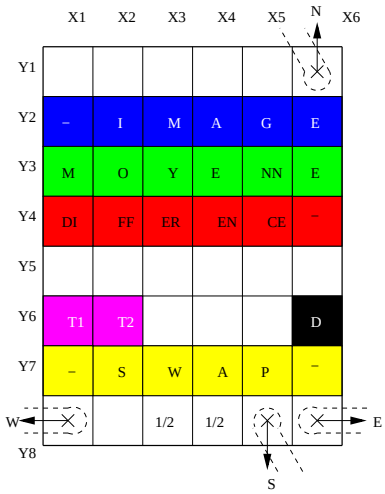



FIG. 2.1 – Utilisation typique de la grille de registres mémoires : exemple d'une détection du mouvement par méthode de soustraction au fond sur 6 bits, en bleu l'image acquise, en vert le fond (calculé à l'aide d'une moyenne récursive par exemple), en rouge la différence au fond, en noir le résultat de la détection binaire (différence seuillée). En jaune, l'occupation d'une ligne de registres pour les calculs ("*swap*") et en mauve deux registres servant de variables temporaires ($T1$ et $T2$) .

| **UNAIRE** | |
|---|---|
| INSTRUCTION | CODE |
| Mise à 1 d'un registre | one |
| Lecture d'un registre | rd |
| Écriture directe | wd |
| Écriture complémentée | wc |
| **BINAIRE** | |
| INSTRUCTION | CODE |
| Mélange de charge (OU logique) | ror |

TAB. 2.1 – Les instructions de l'assembleur rétinien et leur spécificités

Il est à noter que chaque lecture efface la valeur de la case mémoire lue et donc une lecture que l'on veut non destructive doit être nécessairement suivie d'une écriture dans le même registre. Nous pouvons cependant écrire plusieurs fois de suite la même valeur sans la perdre ce qui permet de gagner quelques instructions et d'initialiser rapidement une donnée à l'aide de l'instruction ONE qui met à "1" un registre (remplissant par des "0" ou des "1" tous les bits d'un mot).

L'instruction correspondant au "OU" logique est réalisée par le mélange des charges de deux cellules suivi par un seuillage bas. Ainsi, il suffit que l'une des deux cellules ait une tension suffisamment élevée pour que le seuil requis soit franchi et que l'on obtienne le résultat désiré.

On constate que le fait de devoir "rafraîchir" un registre après l'avoir lu a pour conséquence directe qu'en moyenne, nous effectuons deux fois plus d'opérations d'écriture que de lecture et ceci y compris lors des calculs, puisque le mélange de charge nécessite une lecture des opérandes. En remarquant que l'opération de lecture demande en réalité 6 cycles d'horloge et que celle d'écriture seulement 3, nous comprenons l'importance de rassembler de façon optimale les écritures successives en vue de futurs calculs.

Une autre contrainte liée à l'architecture de Pvlsar34 est qu'on ne peut réaliser les opérations de mélanges de charges que sur la même ligne du bus. Nous ne pouvons donc effectuer un "OU" logique qu'entre deux registres mémoires se trouvant sur la même ligne de la grille de mémoire. Nous avons la possibilité en revanche de venir écrire le résultat direct ou complémenté partout où l'on en a besoin. Cette particularité nous a poussé à réserver une ligne entière de la grille pour les calculs. Cette sorte de RAM[3] interne que nous nommons la ligne de "*swap*" (échange) est utilisée, en effet, intensivement lors des calculs nécessaires au bon déroulement de nos algorithmes. La taille de cette RAM (une ligne soit 6 registres mémoire) est bien dimensionnée car comme nous le verrons lors de l'évaluation des performances des fonctions élémentaires et des algorithmes, elle rend possibles les calculs requis.

Exemple de code rétinien : déplacement de la valeur en Y2 ;X6 du pixel $P(x, y)$ vers la case mémoire Y2 ;X5 du pixel $P(x + 1, y - 1)$ (P.NE). On note Yi ;Xj, la case mémoire se trouvant à la ième colonne et la jème ligne :

```
rd y2 x6;   //lecture de Y2;X6
wd y2 x6;   //sauvegarde
wd y8 x5;   //ecriture en P.S
```

[3] *Random Access Memory* ou mémoire vive par opposition à **ROM** pour *Read Only Memory* où l'on stocke mes données (bien que ici aucun registre ne soit accessible en lecture seulement)

```
rd y1 x6;  //lecture de P.N
wd y8 x1    //ecriture en P.W
rd y8 x6;   //lecture de P.NE
wd y2 x5;   //ecriture de P.NE en Y2;X5
```

L'écriture complémentée nous donne le moyen de réaliser un "NON" logique utile en complément du "OU" logique dont nous disposons afin d'obtenir toutes les autres fonctions logiques, grâce aux lois de De Morgan (Définition 1).

**Définition 1 (Loi de De Morgan)**

$$\begin{aligned} A \vee B &= \overline{\bar{A} \wedge \bar{B}} \\ A \wedge B &= \overline{\bar{A} \vee \bar{B}} \end{aligned}$$

**Définition 2 (Calcul du OU exclusif)**

$$\begin{aligned} A \oplus B &= (\bar{A} \wedge B) \vee (A \wedge \bar{B}) \\ &= \overline{\overline{(\bar{A} \wedge B)} \wedge \overline{(A \wedge \bar{B})}} \end{aligned}$$

Pour mener à bien le calcul du "OU exclusif" sur la rétine (détaillé dans la Table 2.2) nous avons besoin d'effectuer 12 opérations dont 2 lectures, 7 écritures et 3 mélanges de charges (57 cycle d'horloges), 4 registres mémoires sont utilisés par pixel. D'autres exemples d'optimisations du code rétinien se trouvent dans le Chapitre 6 traitant des l'aide à la génération de code pour rétine artificielle.

### 2.2.1 Acquisition et codage

Le processus d'acquisition sur la rétine produit une image binaire représentant les niveaux d'éclairement des pixels vis-à-vis d'un seuil prédéfini. Dans le circuit, nous utilisons une **photodiode** préchargée à une certaine tension, que nous laissons ensuite se décharger. La tension aux bornes de la photodiode va décroître de manière linéaire, à une vitesse dépendante du débit de photons captés. Nous obtenons par conséquent une image binaire de l'éclairement de la scène en mesurant si la tension aux bornes de la photodiode au bout d'un temps de référence $t_{ref}$ est inférieure à une certaine tension de seuil $V_s$.

Ce simple procédé de seuillage peut être réitéré en utilisant plusieurs temps de référence afin d'obtenir une image en niveau de gris. En effet, il suffit d'être capable de mesurer la

| |
|---|
| Copie de $A$ en y7 ;x1 et $A$ en y7 ;x3 |
| **rd y6 x1 ; wd y7 x1 ;** |
| **wc y7 x3 ;** |
| Copie de $B$ en y7 ;x2 et $\bar{B}$ en y7 ;x4 |
| **rd y6 x2 ; wd y7 x2 ;** |
| **wc y7 x4 ;** |
| Calcul de $C = A \wedge \bar{B} = \overline{\bar{A} \vee B}$ en y7 ;x2 |
| **ror y7 x3 etx2 ;** |
| **wc y7 x2 ;** |
| Calcul de $D = \bar{A} \wedge B = \overline{A \vee \bar{B}}$ en y7 ;x1 |
| **ror y7 x1 etx4 ;** |
| **wc y7 x1 ;** |
| Calcul de $C \vee D = A \oplus B$ en y7 ;x1 |
| **ror y7 x1 etx2 ;** |
| **wd y6 x3 ;** |

TAB. 2.2 – Code rétinien correspondant du calcul d'un "OU" exclusif entre deux registres mémoire $A$ en y6 ;x1 et $B$ en y6 ;x2, le résultat en y6 ;x3.

tension aux bornes de la photodiode sans perturber la décharge de celle-ci. Nous pouvons alors effectuer $n$ lectures à des temps de référence $t_1, ..., t_n$. Le premier instant $t_i$ où la tension de la photodiode sera passée en dessous de la tension $V_s$ donnera le niveau de gris normalisé $1 - \frac{i}{n}$ (Figure 2.2).

Une image en niveau de gris est ainsi représentée par un ensemble de coupes binaires, une suite croissante de $n$ images binaires, représentant $n$ seuillages successifs. Finalement, le codage Analogique / numérique du niveau de gris consiste simplement à réaliser, pour chaque pixel, un comptage sur $log(n)$ bits du nombre de coupes où le pixel vaut 1.

La technique de comptage la plus rapide sur le circuit consiste à utiliser un **compteur de Gray**[**?**]. En effet, le principe d'un tel compteur est que les codages de deux nombres consécutifs ne différent que d'un seul bit et donc chaque incrément ne coûte qu'un seul "OU" exclusif (Table 2.3 et Figure 2.3).

**Définition 3 (compteur de Gray)** *Si l'image est représentée par ses coupes* $(C_i)_{1 \leq i \leq n-1}$ *et si l'on note* $Z(p)$ *la plus grande puissance de 2 qui divise un naturel* $p$*, le codage de Gray du niveau de gris est défini par ses chiffres binaires* $(G_i)_{0 \leq i \leq log(n-1)}$ *avec :*

$$G_j = \bigoplus_{Z(i)=2^j} C_i$$

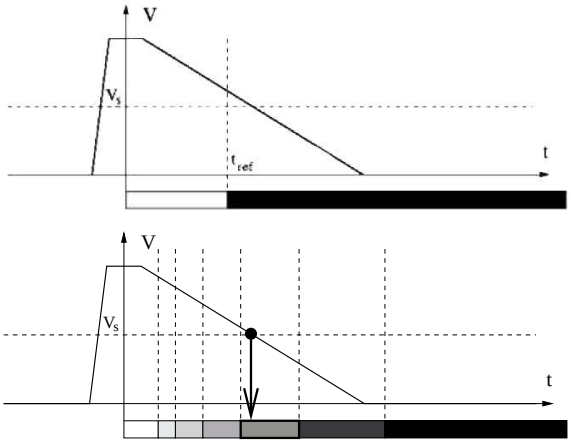
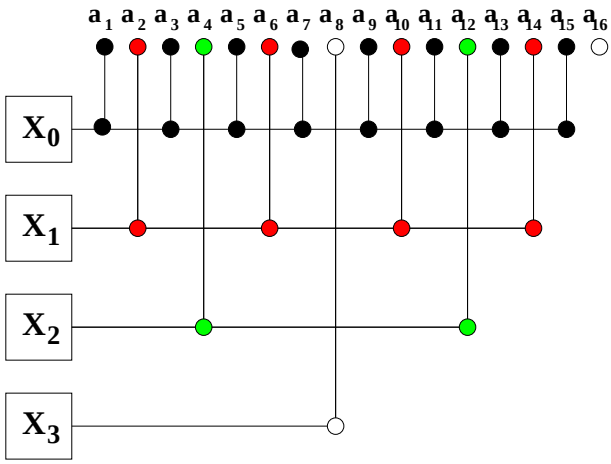



FIG. 2.2 – Acquisition binaire et acquisition d'un niveau de gris par la lecture à différents instants au cours de la décharge de la photodiode.

| Chiffre | Code naturel | Code Gray |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

TAB. 2.3 – Le codage de Gray sur 4 bits.




Nous en déduisons les formules suivantes :

$$
\begin{aligned}
x_0 &= a_1 \oplus a_3 \oplus a_5 \oplus a_7 \oplus a_9 \oplus a_{11} \oplus a_{13} \oplus a_{15} \\
x_1 &= a_2 \oplus a_6 \oplus a_{10} \oplus a_{14} \\
x_2 &= a_4 \oplus a_{12} \\
x_3 &= a_8
\end{aligned}
$$

FIG. 2.3 – Schéma de l'algorithme de l'acquisition en codage de Gray sur 4 bits. Les cercles de couleurs représentent des "OU" exclusifs.

*Où $\bigoplus$ représente l'imparité (OU exclusif).*

Le code Gray utilise seulement $log(n) - 1$ OU exclusifs et pas d'autres registre mémoires que ceux utilisés pour le comptage. Le code naturel est défini par ses chiffres binaires $(n_j)_{0 \leq j \leq log(n)-1}$ tels que les valeurs du niveau de gris représenté est $\sum_{j=0}^{log(n)-1} 2^{n_j}$. On peut passer facilement d'un codage Gray au codage naturel comme le montre la Table 2.4. Le code naturel est ainsi obtenu en $log(n) - 1$ opérations sans utilisation de registres mémoire supplémentaires.

```
//Acquisition de l'image et codage en code Gray
Pour (i = 1; i < n; i++)
{
        //Seuillage du niveau i
        X_i = ACQ(i)
        //Calcul de l'index : plus grande puissance de 2 qui divise i
        index = 0
        x = i
        Tantque ((x%2) == 0) faire {index++; x = x/2}
        G_index = XOR(G_index, X_i)
}
//Passage en code naturel
N_log(n) = G_log(n);
Pour (i = log(n) - 1; i ≥ 0; i--)
        X_i = XOR(G_i, X_i+1);
```

TAB. 2.4 – Algorithme d'acquisition d'une image en niveau de gris.

L'algorithmique intervient ainsi précocement dans les rétines programmables, permettant naturellement d'effectuer des traitements pendant l'acquisition de l'image, en réalisant des calculs à chaque ensemble de niveaux. Un cadre formel plus adapté est donc celui des opérations morphologiques, grâce à l'équivalence entre morphologie en niveaux de gris et morphologie ensembliste sur chaque ensemble de niveau. En outre, nous intervenons sur la dynamique de l'image en modifiant les instants de lecture de la photodiode (compression logarithmique par exemple) ou encore en adaptant les instants de lecture à la répartition des niveaux de gris (i.e. égalisation d'histogramme).

### 2.2.2 Définitions

Dans cette partie, nous introduisons les concepts fondamentaux des outils que nous utilisons sur la rétine numérique programmable. Tout au long de notre présentation, nous utilisons indifféremment deux types de notation pour les opérations élémentaires : ensembliste ou booléen, en fonction de nos besoins. Le Tableau 2.5 explicite l'équivalence de ces deux formalismes.

| FORMALISME | |
|---|---|
| BOOLÉEN | ENSEMBLISTE |
| IMAGE | |
| $p \in \{0,1\}^{\mathbb{Z}^2}$ | $I \in \mathcal{P}(\mathbb{Z}^2)$ |
| $I = P^{-1}(1)$; $p = 1_I$ | |
| OPÉRATIONS DE BASE | |
| *disjonction* | |
| OU logique | Union |
| $p_1 \vee p_2$ | $I_1 \cup I_2$ |
| *conjonction* | |
| ET logique | Intersection |
| $p_1 \wedge p_2$ | $I_1 \cap I_2$ |
| *complémentation* | |
| NON logique | Complémentarité |
| $\bar{p}$ | $\neg I$ |
| *disjonction exclusive* | |
| XOR logique | Différence symétrique |
| $p_1 \triangle p_2$ | $I_1 \Delta I_2$ |
| *différence* | |
| ET NON logique | Différence |
| $p_1 \bar{\rightarrow} p_2$ | $I_1 \setminus I_2$ |

TAB. 2.5 – Correspondance entre notations booléennes et ensembliste .

**Propriété 1** *Une application* $\Phi(E) \rightarrow P(E)$ *est dite :*

*Croissante :*

$$X \subset Y \Rightarrow \Phi(X) \subset \Phi(Y)$$

*Extensive :*

$$X \subset \Phi(X)$$

*Contractante ou anti-extensive :*

$$\Phi(X) \subset X$$

*Idempotente :*

$$\Phi(\Phi(X)) = \Phi(X)$$

$\Psi$ *est dual de* $\Phi$ *si et seulement si :*

$$\Psi(\neg X) = \neg\Phi(X)$$

*où* $\neg$ *représente la complémentarité.*

**Définition 4** *Le translaté de A par x est défini par :*

$$A_x = \{y | \exists a \in A, y = a + x\}$$

### 2.2.3 Topologies dans la maille carrée

Nous considérons uniquement le cas de la maille carrée pour la simple raison que les pixels de la rétine nmérique sont carrés. Il existe cependant d'autres types de topologies dont la maille hexagonale abondamment utilisées en morphologie mathématique.

Soit $\mathbb{Z}^2$ le plan discret. Nous notons $p_1 = P(x_1, y_1)$ et $p_2 = P(x_2, y_2)$ deux points de $\mathbb{Z}^2$ représentés par leurs coordonnées cartésiennes. Nous définissons les relations de voisinage dans la maille carrée de la façon suivante :

**Définition 5** $p_1$ *et* $p_2$ *sont 4-voisins si et seulement si :*

$$\delta_4(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2| \leq 1.$$

$p_1$ *et* $p_2$ *sont 8-voisins si et seulement si :*

$$\delta_8(p_1, p_2) = max(|x_1 - x_2|, |y_1 - y_2|) \leq 1.$$

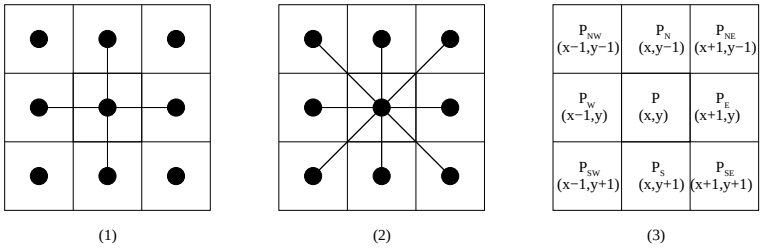



FIG. 2.4 – Les deux relations classiques de connexité dans la maille carrée. (1) 4-connexité (2) 8-connexité (3) noms utilisés pour désigner les 8 voisins d'un point $P$ et leur coordonnées cartésiennes .

La figure 2.4 illustre ces relations de voisinage. Les chiffres 4 et 8 font référence aux nombres de voisins pour chaque relation. On utilisera pour désigner les voisins d'un point $P$ l'indexation par la direction cardinale, comme indiqué sur la figure 2.4.(3).

Les opérateurs $\delta_4$ et $\delta_8$ de la définition 5 sont des distances que l'on dit induites respectivement par la 4- et la 8-connexité.

**Définition 6** *Soit $A$ et $B$ deux sous-ensembles de $\mathbb{Z}^2$. $A$ et $B$ sont dits K-voisins (K=4 ou 8) si et seulement si :*
*Il existe $x \in A$ et $y \in B$ tels que $x$ et $y$ sont K-voisins.*

**Définition 7** *Soit $X \subset \mathbb{Z}^2$, $X$ est une K-composante connexe (K-cc) si et seulement si : Il n'existe pas de partition de $X$ en deux sous-ensembles qui ne soient pas K-voisins. Autrement dit, tous les points de $X$ sont connectés en K-topologies.*

**Définition 8** *Soit $x \in X \subset \mathbb{Z}^2$, Le point $x$ est dit K-intérieur à $X$ si et seulement si tous les K-voisins de $x$ appartiennent à $X$.*

**Fonction distance**

Soit $x \in \mathbb{Z}^2$ et $r \in \mathbb{N}$. Nous notons $\mathcal{B}_K(x,r) = \{y \in \mathbb{Z}^2, \delta_K(x,y) \leq r\}$ la boule de centre $x$ et de rayon $r$ de la distance $\delta_K$.

**Définition 9** *Soit $X \subset \mathbb{Z}^2$.*

*La boule $\mathcal{B}_K(x,r)$ est dite maximale pour $X$ si et seulement si :*

$$\forall y \in X, \forall q \in \mathbb{N}, \mathcal{B}_K(x,r) \subset \mathcal{B}_K(y,q) \Rightarrow (y,q) = (x,r).$$

**Définition 10** *Soit $X \subset \mathbb{Z}^2$.*
*$S_K$ l'axe médian de $X$ relativement à la distance $\delta_K$ est défini comme la réunion des centres de boules maximales de la distance $\delta_K$.*

Cette définition s'avère très importante pour le calcul des squelettes et leur utilisation dans un contexte de recherche de primitives de calcul pour le suivi du mouvement (Chapitre 4 et 5).

**Définition 11** *Soit $X \subset \mathbb{Z}^2$ et $K \in 4, 8$.*
*Nous notons $X^c = \mathbb{Z}^2 \setminus X$ le complémentaire de $X$.*
*La fonction K-distance associée à $X$ est la fonction :*

$$\begin{aligned} \Phi_K : \mathbb{Z}^2 &\rightarrow \mathbb{N} \\ x &\mapsto \Phi_K(x) = \delta_K(x; X^c). \end{aligned}$$

**Propriété 2** *$S_K = \{x \in X; \forall y \in \mathcal{B}_K(x,1), \Phi_K(y) \leq \Phi_K(x)\}$ :*
*La réunion des centres des K-boules maximales est égale à l'ensemble des maxima locaux de la fonction K-distance.*

**Connexité**

**Définition 12 (Nombre de connexité)** *Soit $x$ un point, on note $x_k$ le voisin de $x$ (en partant du voisin nord ($x_0$) et en tournant autour de $x$ dans le sens des aiguilles d'une montre).*
*On définit pour la 8-connexité*

$$\mathcal{N}_{c8}(x) = \sum_{i=0}^{3} (x_{2i+1} \vee x_{2i+2}) \wedge \overline{x_{2i}}$$

*et pour la 4-connexité*

$$\mathcal{N}_{c4}(x) = \sum_{i=0}^{3} (\overline{x_{2i+1} \wedge x_{2i+2}}) \wedge x_{2i}$$

*[YTF75]*

Le nombre de K-connexité $\mathcal{N}_{cK}(x)$ est exactement égal au nombre de K-cc de $V_8(x) \setminus \{x\}$, sauf pour les points K-intérieurs. Par exemple, un point $X$ isolé ou un point intérieur aura un nombre de connexité $\mathcal{N}_{c8}(x) = 0$ et $\mathcal{N}_{c4}(x) = 0$.

**Propriété 3** *Un point $x$ est K-simple si et seulement si $\mathcal{N}_{cK}(x) = 1$.*

Dans le plan $\mathbb{R}^2$ muni de la topologie euclidienne, dire qu'une composante connexe $A$ de $X$ "entoure" une composante connexe $B$ de $X^c$ (ou inversement) signifie qu'il existe une *courbe simple fermée* (**courbe de Jordan**) $\gamma$ appartenant à $A$ telle que $B$ appartient à l'*intérieur* de $\gamma$. On utilise implicitement le **théorème de Jordan**, qui dit que toute courbe de Jordan sépare le plan en deux parties distinctes, une finie (qu'on appelle intérieur) et une infinie (l'extérieur).

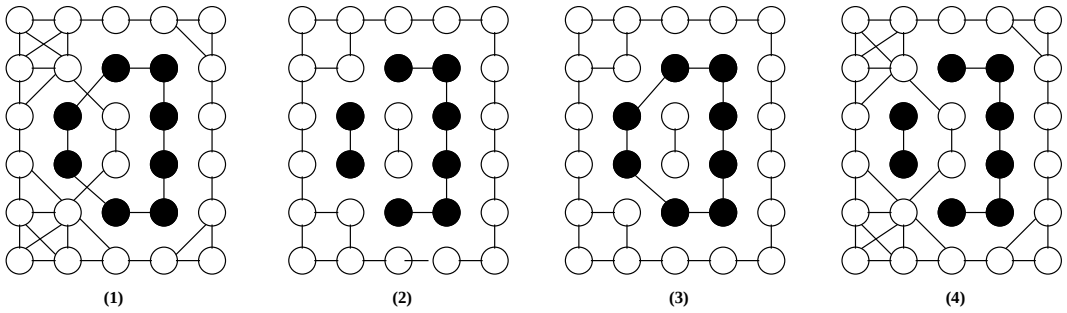



FIG. 2.5 – Le théorème de Jordan dans la maille carrée n'est valable que si l'on choisit des connexités différentes pour $X$ et pour $X^C$. (1) 8-connexité pour $X$ et $X^C$ (2) 4-connexité pour $X$ et pour $X^C$ (3) 8-connexité pour $X$ et 4-connexité pour $X^C$ (4) 4-connexité pour $X$ et 8-connexité pour $X^C$

Or, dans le plan discret, la maille carrée, cette propriété n'est vraie que dans des conditions particulières. Dans la Figure 2.5, si l'on considère la 8-connexité, l'objet $X$, en noir, est une courbe de Jordan, mais $X^c$ forme une unique composante connexe (1). En 4-connexité (2), au contraire, $X$ n'est pas une courbe de Jordan, mais est formé de plusieurs composantes connexes. Cependant, il sépare $X^c$ en deux composantes connexes. En fait, le problème est résolu en choisissant pour $X$ et pour $X^c$ deux types de connexité différentes. Ainsi, dans la maille carrée, on considérera toujours soit la (8,4)-topologie (3), soit la (4,8)-topologie (4).

Soit $x \in X$. Si $V_K(x)$ est l'ensemble des K-voisins de $x$ et $\bar{\text{K}}$ la connexité duale[4], alors nous avons le résultat suivant :

[4] $\bar{K} = 12 - K$.

**Théorème 1** *Soit $x \in X$ tel que $V_{\bar{K}}(x) \cap X^c \neq \emptyset$, alors $x$ est K-simple si et seulement si $x$ est K-voisin d'exactement une K-cc de $(V_8(c) \setminus \{x\}) \cap X$.*

Si $x$ est K-simple alors $x$ est nécessairement un point du $\bar{\text{K}}$-contour[5], nous évitons ainsi que la destruction de $x$ crée une nouvelle $\bar{\text{K}}$-cc de $X^c$ (un trou). S'il existe au moins deux K-cc dans $(V_8(x)\setminus\{x\})\cap X$, alors soit ces deux K-cc appartiennent au niveau global à deux K-cc différentes et alors la destruction de $x$ déconnecte K, soit elle connecte deux $\bar{\text{K}}$-cc de $X^c$ (perce un trou). Dans les deux cas la destruction de $x$ entraîne bien un changement de topologie.

## 2.3 Les bases du calcul sur rétine numérique

Nous décrivons dans la suite les fonctions élémentaires que nous utilisons sur la rétine pour l'élaboration de nos algorithmes dans le cadre de la détection et l'estimation du mouvement. Pour chaque fonction nous indiquons le pseudo-code associé et les améliorations que nous apportons au code rétine afin d'utiliser au mieux les spécificités du circuit rétinien. Nous notons aussi les performances des fonctions décrites en termes de temps de calcul (nombres d'instructions) et d'utilisation de la mémoire (nombre de registres nécessaires).

Les deux premières fonctions arithmétiques essentielles au calcul sur la rétine sont la soustraction et l'addition. Nous verrons ensuite comment la soustraction joue le double rôle d'opération arithmétique et de comparateur binaire.

### 2.3.1 Addition et soustraction

Soit $A = \{A_1, ..., A_n\}$, $B = \{B_1, ..., B_n\}$ et $S = \{S_1, ..., S_n\}$ trois mots tels que $S = A - B$ d'une part et $S = A + B$ d'autre part et $C_{in}$ et $C_{out}$ les retenues entrante et sortante (Figure 2.6). La Table 2.6 nous donne respectivement les tables de vérité pour les fonctions de soustraction et d'addition.

Les variables $C_{in}$ et $C_{out}$ sont les retenues entrante et sortante et représentent, à la fin du calcul le bit de signe du résultat. De plus, les nombres négatifs sont codés en compléments à 2, c'est à dire que le bit de poids fort du mot représente le signe de celui-ci.

[5]C'est-à-dire que $x$ a un $\bar{\text{K}}$-voisin dans $X^c$.

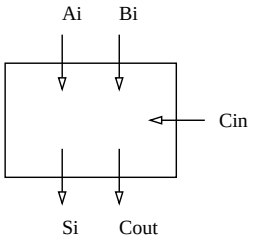



FIG. 2.6 – Schéma des opérations d'addition et de soustraction .

| $A_i$ | $B_i$ | $C_{in}$ | $S_i$ | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

(1)

| $A_i$ | $B_i$ | $C_{in}$ | $S_i$ | $C_{out}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(2)

TAB. 2.6 – Table de vérité de la soustraction (1) et de l'addition (2).

Nous utilisons un tableau de Karnaugh pour déduire les expressions logiques et nous remarquons que la formule $S_i = A_i \oplus B_i \oplus C_{in}$ se trouve dans les deux cas ; seul le calcul de la retenue sortante change. Nous obtenons $C_{out} = B_i C_{in} + \bar{A}_i C_{in} + \bar{A}_i B_i$ pour la soustraction et $C_{out} = B_i C_{in} + A_i C_{in} + A_i B_i$ pour l'addition. Dans la pratique, nous utilisons de préférence les formules $C_{out} = (A_i \bar{\oplus} B_i) C_{in} + \bar{A}_i B_i$ et $C_{out} = (A_i \oplus B_i) C_{in} + A_i B_i$. Ce qui nous donne les pseudo-codes suivants de la Table 2.7.

| | |
|---|---|
| $D = A_i \oplus B_i$ | $D = A_i \oplus B_i$ |
| $E = \bar{A}_i \wedge B_i$ | $E = A_i \wedge B_i$ |
| $S = D \oplus C_{in}$ | $S_i = D \oplus C_{in}$ |
| $D = \bar{D} \wedge C_{in}$ | $D = D \wedge C_{in}$ |
| $C_{out} = D \vee E$ | $C_{out} = D \vee E$ |
| (1) | (2) |

TAB. 2.7 – Pseudo-codes de la soustraction (1) et de l'addition (2).

Il est possible d'effectuer ces calculs de manière plus économique en rassemblant des opérations semblables et en utilisant l'expression du "OU" exclusif suivante :

$$A \oplus B = (\bar{A} \wedge B) \vee (A \wedge \bar{B}) = (\overline{A \wedge B}) \vee (\overline{\bar{A} \wedge \bar{B}})$$

Finalement, en ne prenant en compte que des "OU" logiques, nous obtenons les algorithmes de la Table 2.8.

| | |
|---|---|
| $E = \overline{\bar{A}_i \vee B_i}$ | $E = \overline{\bar{A}_i \vee \bar{B}_i}$ |
| $F = \overline{A_i \vee \bar{B}_i}$ | $F = \overline{A_i \vee B_i}$ |
| $D = \overline{E \vee F} = A_i \oplus B_i$ | $D = \overline{\overline{E \vee F}} = A_i \oplus B_i$ |
| $F = \overline{D \vee \overline{C_{in}}}$ | $F = \overline{\bar{D} \vee \overline{C_{in}}}$ |
| $G = \overline{\bar{D} \vee C_{in}}$ | $G = \overline{D \vee C_{in}}$ |
| $S_i = F \vee G = A_i \oplus B_i \oplus C_{in}$ | $S_i = \overline{F \vee G} = A_i \oplus B_i \oplus C_{in}$ |
| $C_{out} = D \vee E = ((A_i \bar{\oplus} B_i) \wedge C_{in}) \vee (A_i \wedge \bar{B}_i)$ | $C_{out} = D \vee E = ((A_i \oplus B_i) \wedge C_{in}) \vee (A_i \wedge B_i)$ |
| (1) | (2) |

TAB. 2.8 – Pseudo-codes optimisés de la soustraction (1) et de l'addition (2).

Code Rétinien pour le calcul d'une soustraction entre deux registres mémoire, $A_i$ se trouve en Y7 ;X1, $\bar{A}_i$ en Y7 ;X2, $B_i$ en Y7 ;X3 et $\bar{B}_i$ en Y7 ;X4. Le résultat est écrit en Y7 ;X1 et la retenue se trouve en Y6 ;X1 :

```
void Arithmetique_Bit(int op)
{

  if (op==0)//Soustraction
  {

  // Calcul de B = X_i et non(M_i) sur y7;x1 et y7;x4
    ror y7 x1 etx4;
    wc y7 x1;
    wc y7 x4;
  // Calcul de non(X_i) et M_i sur y7;x2
    ror y7 x2 etx3;
    wc y7 x2;
  // Calcul de A=X_i xor M_i en y7;x1 et non(A) en y7;x2
    ror y7 x1 etx2;
    wd y7 x1;
    wc y7 x2;
  // Copie de S en y7;x3 et non(S) en y7;x5
    rd y6 x1;
    wd y7 x3;
    wc y7 x5;
  // Calcul de C = non(A) et S y7;x1 et non(C) en y7;x5
    ror y7 x1 etx5;
    wc y7 x1;
    wc y7 x5;
  // Calcul de A et non(S)  en y7;x2
    ror y7 x2 etx3;
    wc y7 x2;
  // Calcul de X_i = A xor S en y7;x1
    ror y7 x1 etx2;
    wd y7 x1;
  // Mise a jour de S = B ou non(C) en y6;x1
    ror x7 x4 etx5;
    wd y6 x1;
```

```
  }else{//Addition

  // Calcul de B = X_i et M_i sur y7;x2 et en y7;x5
    ror y7 x2 etx4;
    wc y7 x2;
    wc y7 x4;
  // Calcul de non(M_i) et non(X_i) sur y7;x1
    ror y7 x1 etx3;
    wc y7 x1;
  // Calcul de A=X_i xor M_i en y7;x1 et non(A) en y7;x2
    ror y7 x1 etx2;
    wc y7 x1;
    wd y7 x2;
  // Copie de S en y7;x3 et non(S) en y7;x5
    rd y6 x1;
    wd y7 x3;
    wc y7 x4;
  // Calcul de C = A et S en y7;x2 et non(C) en y7;x5
    ror y7 x2 etx5;
    wc y7 x2;
    wc y7 x5;
  // Calcul de non(A) et non(S) en y7;x1
    ror y7 x1 etx3;
    wc y7 x1;
  // Calcul de X_i = A xor S en y7;x1
    ror y7 x1 etx2;
    wc y7 x1;
  // Mise a jour de S = non(B) ou non(C)
    ror x7 x4 etx5;
    wd y6 x1;

  }

}
```

Les opérations arithmétiques de base nécessitent donc, par bit, 20 opérations rétiniennes, dont 8 lectures ce qui correspond à 96 cycles d'horloge. Nous utilisons pour ces opérations 6 registres mémoire, dont une variable hors de la ligne de swap qui correspond à la retenue propagée de bit en bit.

### 2.3.2 Valeur Absolue

Nous calculons la valeur absolue de $X$ en utilisant le signe obtenu lors de la soustraction correspondant à la valeur de la retenue sortante $C_{out}$ issue du bit de poids fort. L'algorithme consiste à parcourir tous les bits en partant du bit de poids faible, à laisser inchangés tous les bits jusqu'au premier "1" rencontré puis à inverser tous les bits après. Nous avons donc besoin d'une bascule nous permettant de savoir si nous avons déjà rencontré le premier "1" du mot à inverser (si le bit de signe est à 1). Nous l'initialisons tout simplement par un "ET" logique entre le bit de signe et le bit du poids faible, lequel ne sera jamais inversé quel que soit le signe (Table 2.9).

$B = S \wedge X_0$
**Pour (i=1 ; i≤n-1 ; i++) {**
  $X_i = X_i \oplus B$ ;
  $A = S \wedge X_i$ ;
  $B = B \vee A$ ;
**}**

TAB. 2.9 – Pseudo-code du calcul de la valeur absolue.

Code Rétinien pour le calcul de la valeur absolue d'un bit du mot $X_i\,(i > 1)$ , $X_i$ se trouve en Y7 ;X1, $\bar{X}_i$ en Y7 ;X2. Le résultat est écrit en Y7 ;X1, le bit de signe $S$ est en Y6 ;X1 et la bascule $B$ se trouve en Y6 ;X2 :

```
//initialisation
// Calcul de B=S et X_0=non(non(S) ou non(X_0)) en y6;x2

//Copie de non(X_0) en y7;x1
rd y1 x6; wd y1 x6;
wc y7 x1;
//Copie du Bit de signe S en y6;x1 et non(S) en y7;x2
rd y6 x1; wd y6 x1;
wc y7 x2;
//Calcul de B en y6;x2
ror y7 x1 etx2;
wc y6 x2;

void Valeur_Absolue_Bit()
{
// Copie de B sur y7;x3 et y7;x4 et non(B) sur y7;x5
rd y6 x2;
wd y7 x3;
wd y7 x4;
wc y7 x5;
// Calcul de X_i et non(B) sur y7;x2
ror y7 x2 etx3;
wc y7 x2;
// Calcul de non(B) et X_i sur y7;x1
ror y7 x1 etx5;
wc y7 x1;
// Calcul de X_i = B xor X_i
// sur y7;x1 et non(X_i) sur y7;x2
ror y7 x1 etx2;
wd y7 x1;
wc y7 x2;
// copie de non(S) sur y7;x3;
rd y6 x1;wd y6 x1;
wc y7 x3;
// Calcul de A = X_i et S sur y7;x2
ror y7 x2 etx3;
wc y7 x2;
// Mise a jour de B = A ou B
ror y7 x2 etx4;
wd y6 x2;
}
```

Pour chaque bit, nous effectuons pour le calcul de la valeur absolue 18 opérations dont 7 de lectures et 11 d'écritures, soit 87 cycles d'horloges. Nous utilisons 7 registres mémoire dont 2 bits hors de la ligne de swap (un pour le bit de signe et le second pour la bascule).

### 2.3.3 Comparateur

Le comparateur bit à bit nous permet de savoir si un mot est strictement inférieur ou supérieur à un autre. Pour une comparaison large, il est moins coûteux de faire une soustraction et de considérer le bit de signe. L'égalité est déduite de la différence entre les deux mots ou testée ultérieurement. En effet, nous testons l'égalité par les formules suivantes : soit deux mots de $n$ bits $A = \{A_1, ..., A_n\}$ et $B = \{B_1, ..., B_n\}$ tel que $S = \{S_1, ..., S_n\} = A - B$, $A = B$ si et seulement si $\bigvee_{i=1}^{n} S_i = 0$. Ou encore si et seulement si $\bigwedge_{i=1}^{n} A_i \oplus B_i = 0$. A noter que cette deuxième option permet de ne pas effectuer de soustraction mais nous oblige à effectuer $n$ "OU" EXCLUSIF, opération assez coûteuse. Nous privilégions donc de tester l'égalité après avoir rencontré une soustraction. Nous arrangeons donc les algorithmes pour se trouver dans cette situation, par exemple en modifiant l'ordre des instructions. Cette astuce est souvent employée dans nos algorithmes comme nous le verrons par la suite.

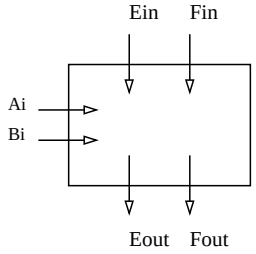



FIG. 2.7 – Schéma de l'opération de comparaison .

D'un point de vue logique, cet opérateur comporte donc quatre entrées, les deux bits courants des mots à comparer et deux bascules correspondant aux expressions logiques suivantes : $[(A_i > B_i)\&(E \neq 1)]$ et $[(A_i < B_i)\&(F \neq 1)]$ (Figure 2.7).

On a $E = 1$ si et seulement si $A < B$ et $F = 1$ si et seulement si $A > B$. Le cas $E = 1$ et $F = 1$ est donc impossible dans l'algorithme et on ne le considère pas dans la Table 2.10. De même, si $A_i = B_i$ et $E = 1$ ou $F = 1$ alors rien ne se passe car soit $A_i$ et $B_i$ sont

| $E_{in}$ | $F_{in}$ | $A_i$ | $B_i$ | $E_{out}$ | $F_{out}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | * | * | 1 | 0 |
| 0 | 1 | * | * | 0 | 1 |

TAB. 2.10 – Table de vérité du comparateur bit à bit, '*' représente une valeur quelconque.

égaux soit $E$ ou $F$ ont déjà basculé.

On déduit comme précédemment le pseudo-code du comparateur, voir Table 2.11.

**Pour (i=7 ; i≥0 ; i- -) {**

$$
\begin{aligned}
M &= A_i \wedge \bar{B}_i \\
N &= \bar{A}_i \wedge B_i \\
O &= N \wedge \bar{D} \\
E_{out} &= E_{in} \vee O \\
O &= M \wedge \bar{D} \\
F_{out} &= F_{in} \vee O \\
M &= M \vee N \\
D &= D \vee M
\end{aligned}
$$

**}**

TAB. 2.11 – Pseudo-code du calcul de la comparaison entre deux mots $A$ et $B$.

Code Rétinien pour le calcul de la comparaison de deux bits des mot $A$ et $B$, $A_i$ se trouve en Y7 ;X1 $\bar{A}_i$ en Y7 ;X2, et $B_i$ se trouve Y7 ;X3 et $\bar{B}_i$ en Y7 ;X4. Le résultat est écrit en Y7 ;X1, les bascules $D$, $E$ et $F$ se trouvent respectivement en Y6 ;X1, Y6 ;X2 et Y6 ;X3 :

```
void Compare_Bit()
{
// Calcul du OU dans A (A = M_i et non(X_i)), sur y7;x1,
// et de non(A) sur y7;x4
ror y7 x1 etx4;
wc y7 x1;
wd y7 x4;
// Calcul du OU dans B (B = X_i et non(M_i)), sur y7;x2,
// et de non(B) sur y7;x3
ror y7 x2 etx3;
wc y7 x2;
wd y7 x3;
// Ecriture de D sur y7;x5 et sur y7;x6
rd y6 x1; wd y6 x1;
wc y7 x5;
wc y7 x6;
// Calcul de G = A et non(D) sur y7;x4
ror y7 x4 etx5;
wc y7 x4;
// Ecriture de E sur y7;x5
rd y6 x2; wd y6 x2;
wc y7 x5;
// Calcul de E = E ou G sur y6;x2
ror y7 x4 etx5;
wc y6 x2;
// Calcul de G = B et non(D) sur y7;x3
rmt1 y7 x3 etx6;
wc y7 x3;
// Copie de F sur y7;x4
rd y6 x3; wd y6 x3;
wc y7 x4;
// Calcul de F = F ou G sur y6;x3
ror y7 x3 etx4;
wc y6 x3;
// Calcul de A = A ou B sur y7;x1
ror y7 x1 etx2;
wd y7 x1;
// Copie de D sur y7;x2
rd y6 x1;
wc y7 x2;
// Calcul de D = D ou A sur y6;x1
ror y7 x1&x2;
wc y6 x1;
}
```

La comparaison nécessite 30 opérations par bit, dont 12 de lectures et 18 d'écritures, soit 144 cycles d'horloge. Elle occupe 8 registres mémoires dont 3 registres hors de la ligne de calcul ("swap") correspondant aux 3 bascules.

### 2.3.4 Détecteur de points isolés binaire

FIG. 2.8 – Configuration schématique d'un point isolé.

Ce petit algorithme permet de détecter des configurations particulières de pixels correspondant à des points isolés (Figure 2.8). En effet, nous utilisons souvent ce détecteur comme filtrage afin de supprimer un bruit temporel. Nous pouvons considérer en effet que la présence d'un point isolé suite à une différentiation temporelle est dûe au bruit. Nous définissons un tel point isolé par :

$$D = P \wedge (\overline{P.N \vee P.NE \vee P.E \vee P.SE \vee P.S \vee P.SW \vee P.W \vee P.NW})$$

A l'aide des formules de Morgan, on réecrit la formule ci-dessus afin de n'exécuter qu'une seule opération par instruction. Ainsi, sur la rétine, on utilise le pseudo-code de la Table 2.12. On effectue 41 opérations rétiniennes pour effectuer une détection de points isolés.

| | | |
|---|---|---|
| $P$ | $=$ | $\bar{P}$ |
| $P_1$ | $=$ | $P.N \wedge P.E$ |
| $P_1$ | $=$ | $P_1 \wedge P_1.SW$ |
| $P_2$ | $=$ | $P.N \wedge P.S$ |
| $P_2$ | $=$ | $P_2.E \wedge P_2.W$ |
| $P_1$ | $=$ | $P_1 \wedge P_2$ |
| $D$ | $=$ | $\bar{P} \wedge P_1$ |
| $P$ | $=$ | $\bar{P}$ |

TAB. 2.12 – Pseudo-code du calcul de la détection de points isolés.

### 2.3.5 Détecteur de "contours" morphologiques binaires

Pour finir avec cet aperçu des fonctions élémentaires, nous décrivons un détecteur de "contours" morphologique qui en fait une variante du détecteur de points isolés précédemment présenté. En fait de contour, il s'agit de point érodé ou plus précisemment les points résultants de l'opération consistant á retirer l'érodé de l'image dans l'image originale et appelé parfois point frontière. On n'obtient donc pas le contour à proprement parler de l'image mais un ensemble de points appartenant au contour. D'un point de vue morphologique, il s'agit du ET logique de l'ensemble des points de l'érodé. Dans l'algorithme de la Table 2.13, la variable $P_1$ nous sert à calculer les points intérieurs, pour les retirer de l'image originale.

| | | |
|---|---|---|
| $P_1$ | $=$ | $P.W \wedge P.E$ |
| $P_1$ | $=$ | $P \wedge P_1$ |
| $P_2$ | $=$ | $P.N \wedge P.S$ |
| $P_2$ | $=$ | $P \wedge P_2$ |
| $P_1$ | $=$ | $P_1.N \wedge P_1.S$ |
| $P_2$ | $=$ | $P_2.W \wedge P_2.E$ |
| $P_1$ | $=$ | $P_1 \wedge P_2$ |
| $D$ | $=$ | $P \wedge \overline{P_1}$ |

TAB. 2.13 – Algorithme de calcul des contours morphologiques.

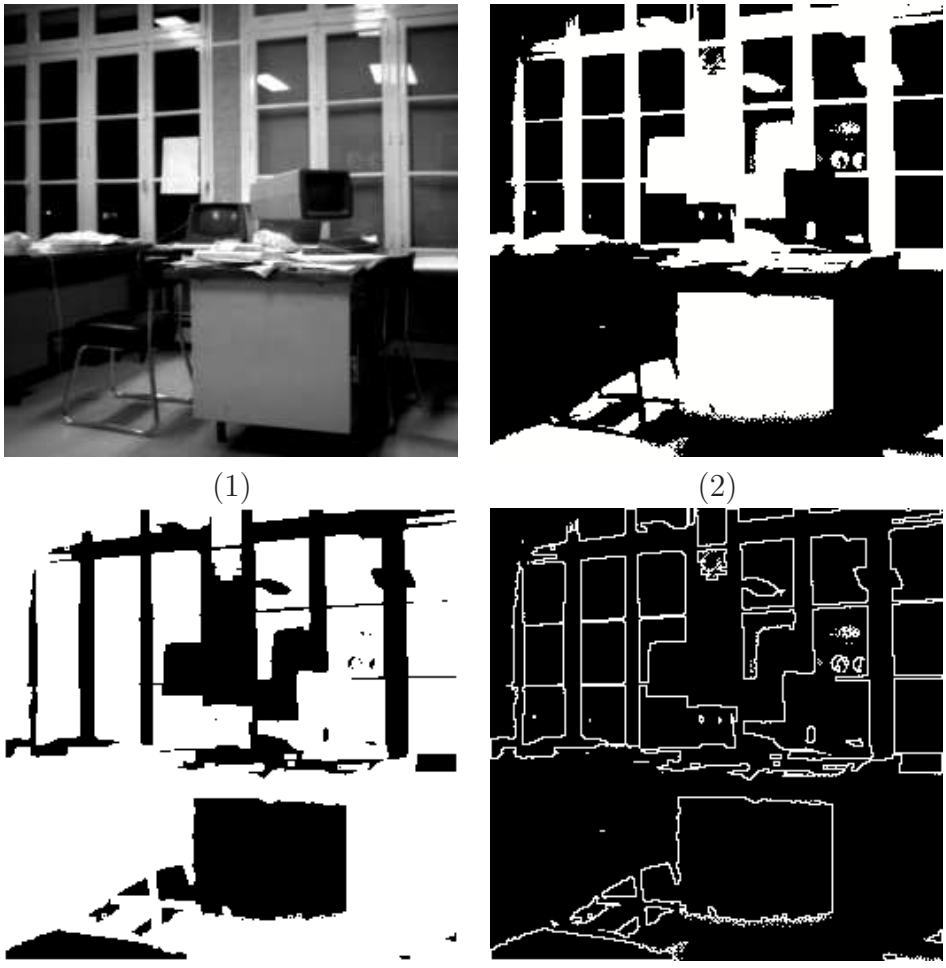
La Figure 2.9 présente le résultat du calcul des contours sur l'image de test "Bureau".

### 2.3.6 L'opérateur point extrême binaire

Nous définissons un point extrême par un point qui n'a qu'un seul voisin dans l'image binaire (Figure 2.10). On parcourt l'ensemble des points qui constitue le voisinage de $P$ et on compte le nombre de points à 1. Si il n'en n'existe qu'un, alors $P$ est extrême. En pratique on utilise deux bascules, la première indique la détection d'un voisin à 1 et la seconde vérifie qu'il n'y en a pas d'autres.

Exemple en 4-connexité

En 4-connexité pour qu'un point $P$ soit extrême, il faut qu'un et un seul de ses voisins directs soit à 1 alors que tous les autres sont à 0. C'est le calcul réalisé par la fonction $f$ suivante :



(1) (2)

(3) (4)

FIG. 2.9 – Exemple de détection de contour morphologique sur l'image de test "Bureau" : (1) image originale ; (2) image seuillée ($\gamma = 100$) ; (3) inversion de l'érosion de l'image seuillée ($\bar{P}_1$) ; (4) contour détectés.

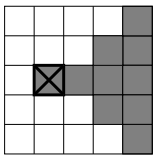


FIG. 2.10 – Exemple de point extrême

$$\begin{aligned} f &= (P.N\overline{P.EP.SP.W}) \vee (\overline{P.N}P.E\overline{P.SP.W}) \vee (\overline{P.NP.E}P.S\overline{P.W}) \vee (\overline{P.NP.EP.S}P.W) \\ &= \overline{P.NP.E}(\overline{P.S}P.W \vee P.S\overline{P.W}) \vee \overline{P.SP.W}(\overline{P.N}P.E \vee P.N\overline{P.E}) \end{aligned}$$

Cette expression se calcule donc en effectuant 12 opérations. A comparer avec l'opérateur de détection de points extrêmes en 8-connexité qui se calcule à l'aide de 20 opérations et nécessite l'utilisation de 5 registres mémoires (Table 2.14) .

| |
|---|
| **Initialisation** |
| $P_1 = P.N$ |
| $P_2 = P.N \wedge P.NE$ |
| **Pour** $(dir = NE$ **;** $dir \leq NW$ **;** $dir++)$**{** |
| $P_1 = P_1 \oplus P_{dir}$ |
| $P_3 = P_1 \wedge P_{dir+1}$ |
| $P_2 = P_2 \vee P_3$ |
| **}** |

TAB. 2.14 – Détecteur de points extrêmes 8-connexe.

## 2.4 Conclusion

Nous avons présenté dans ce chapitre les principes de base du calcul sur rétine numérique programmable et de son algorithmie de traitement d'images. Ces principes s'inscrivent dans le cadre des prétraitements nécessaires à un système de vision.

La Table 2.15 présente les opérateurs binaires introduits dans ce chapitre ainsi que leurs homologues en niveaux de gris.

| Opération | Opérateur binaire | Opérateur en niveaux de gris |
|---|---|---|
| complément | $\neg$ X | $255 - X(t)$ |
| intersection | $X \cap Y$ | $\min_t\{X(t), Y(t)\}$ |
| union | $X \cup Y$ | $\max_t\{X(t), Y(t)\}$ |
| érosion | $X \ominus B$ | $\inf_{y \in B}\{X(t+y) - B(y)\}$ |
| dilatation | $X \oplus B$ | $\sup_{y \in B}\{X(t-y) - B(y)\}$ |
| ouverture | $X \circ B$ | $(X \ominus B) \oplus B$ |
| fermeture | $X \bullet B$ | $(X \oplus B) \ominus B$ |

TAB. 2.15 – Tableau récapitulatif des opérations binaires ayant un homologue en niveaux de gris.

La Table 2.16 nous renseigne sur les performances des fonctions élémentaires présentées en termes de nombre d'instructions et d'occupation de la mémoire. On remarque comme nous l'avions annoncé que l'on effectue 30% d'écritures de plus que de lectures. Toutes ces fonctions, représentent à peu près le même nombres d'instructions et la même quantité de mémoire occupée.

| Opérations | instructions algorithmiques | lecture | écriture | total | coût | occupation |
|---|---|---|---|---|---|---|
| addition | 5 | 8 | 12 | 20 | 96 | 6 |
| soustraction | 5 | 8 | 12 | 20 | 96 | 6 |
| valeur absolue | 3 | 7 | 11 | 18 | 87 | 7 |
| comparaison | 8 | 12 | 18 | 30 | 144 | 8 |
| érosion 4-connexe | 4 | 8 | 12 | 20 | 96 | 8 |
| érosion 8-connexe | 8 | 9 | 14 | 23 | 111 | 7 |
| dilatation 4-connexe | 4 | 8 | 12 | 20 | 96 | 8 |
| dilatation 8-connexe | 8 | 9 | 14 | 23 | 111 | 7 |
| détection de points isolés | 8 | 18 | 23 | 41 | 132 | 9 |

TAB. 2.16 – Tableau récapitulatif des opérations élémentaires disponibles sur la rétine artificielle programmable et évaluation de leurs performances en termes de temps de calcul (nombre d'instructions) et en occupation mémoire (nombre de registres mémoires nécessaires) .

Dans les deux premiers chapitres nous avons posé les bases du concept et des traitements rétiniens. Nous pouvons maintenant présenter les algorithmes d'analyse du mouvement. Les trois chapitres suivants sont respectivement une étude de la détection du

mouvement, des structures spatio-temporelles exploitables comme primitives du mouvement, et enfin de l'estimation du mouvement. Ils constituent le coeur de la thèse et feront abondamment appel à des notions introduites dans ce chapitre.

# Chapitre 3

# Détection du mouvement

## 3.1 Contexte de travail

Afin de comprendre pourquoi nous avons choisi de porter certains types d'algorithmes sur la rétine artificielle, il convient de rappeler notre contexte de travail. Cette étude s'inscrit dans le cadre d'un contrat industriel ayant pour objectif l'élaboration d'un système de télésurveillance en extérieur de longue autonomie ( concept du **capteur abandonné**, un système qui est mis en place puis laissé "à l'abandon" un long moment sans aucune intervention quelconque d'un opérateur). La rétine artificielle est bien adaptée à ce cadre de part sa compacité et sa faible consommation, mais en même temps, elle amène certaines contraintes en terme de capacité de mémorisation. De plus son algorithmique particulière, exposée dans le chapitre précédent (Chapitre 2), nous conduit à employer certains types de méthodes, adaptées à l'architecture de la rétine.

Le défi de la détection du mouvement dans le cas de capteurs fixes tient dans la capacité d'effectuer une bonne segmentation des objets en mouvement indépendamment de leur taille, de leur vélocité, ou de leur contraste par rapport au fond. De plus, l'objectif de nos travaux est la mise en place d'un système capable de s'adapter à toutes les scènes, en particulier dans un contexte de scènes en extérieur qui offrent une plus large gamme de conditions lumineuses et de types de bruits spatiaux et temporels. Notre contexte de travail concerne plutôt les scènes extérieures ; cependant nous ne limiterons pas nos expérimentations à de telles scènes.

Dans notre contexte, les algorithmes de détection du mouvement doivent tenir compte des contraintes suivantes :

- Le système doit être en mesure de fonctionner sans intervention humaine pendant un long moment, et être capable de prendre en compte des changements graduels ou soudains tels que les variations d'illumination ou la présence de nouveaux objets statiques dans la scène. Le système doit donc être **temporellement adaptatif** ;
- Le système doit être capable d'éliminer tout mouvement inintéressant tel que le bruit résultant d'un cours d'eau ou de hautes herbes agitées par le vent. Il doit être robuste à de petits mouvements du capteur. Il doit donc y avoir une estimation **locale** du fond ;
- Le système doit être temps réel[1], compact et basse consommation. Les algorithmes ne doivent pas consommer trop de ressources, en terme de **temps de calcul** et de **capacité mémoire**.

Les deux premières conditions impliquent que les mesures statistiques de l'activité temporelle doivent être effectuées localement en chaque pixel, et constamment remises à jour. Cela exclut toutes les approches utilisant des modèles simples de différence trame à trame, de seuillage global et les méthodes utilisant une moyenne comme seuil de décision (les mouvements induisant une différence supérieure à la moyenne).

Ainsi les techniques de **différence au fond** semblent bien adaptées pour ce type de contexte en choisissant toutefois une méthode qui soit compatible avec les capacités de mémorisation de la rétine. Dans ce type de méthodes, un mouvement est détecté lorsque la différence entre le fond calculé et l'image courante est supérieure à un certain seuil, qui peut-être global ou local et adaptatif. Nous avons donc choisi d'implanter un algorithme à base de moyenne récursive.

En effet, l'utilisation d'une technique basée sur une méthode récursive se justifie par le coût en terme d'occupation mémoire constant tout au long de la détection. Le calcul d'une moyenne récursive ne nécessite pas de conserver en mémoire toutes les valeurs passées dans un tampon mémoire de la longueur de la séquence. Seule la trame courante et la valeur calculée à la dernière trame sont nécessaires.

C'est en cherchant des raffinements pour ces méthodes récursives que nous sommes arrivés à l'élaboration des deux autres algorithmes proposés.

Le premier est issu d'une réflexion sur l'utilisation des opérateurs morphologiques, bien adaptés à la rétine, dans un cadre récursif.

La seconde réflexion porte, elle, sur un opérateur disposant d'une grande adaptativité et permettant de suivre des objets dans plusieurs modes de mouvement. Nous nous sommes alors intéressés à des méthodes qui nous ont conduites à l'élaboration de l'esti-

[1] de l'ordre de 40 images/seconde

mateur Σ-Δ.

A plusieurs reprises dans ce chapitre et dans les chapitres suivants nous présentons des images prises depuis le dispositif expérimental comprenant une rétine artificielle que nous nommons banc algorithmique. Il s'agit d'un ordinateur, servant à programmer et à communiquer les codes sources compilés vers le cortex de la rétine, reliés au couple rétine-cortex. De plus cet ordinateur sert aussi à l'affichage des images résultats par le biais d'une communication via ethernet (voir Chapitre 1 Section 1.2.3 Figure 1.15).

## 3.2 Détection du mouvement sur rétines numériques

Le cadre de travail précédemment présenté permet de justifier le fait que nous n'aborderons pas ici toutes les techniques de détection du mouvement mais celle pouvant faire l'objet d'une étude sur rétine numérique. Nous nous sommes donc résolument concentrés sur des traitements de bas niveau, en faisant abstraction ici de la mise correspondance de haut niveau que nous aborderons plus loin (Chapitre 5). E, effet, les deux approches sont sensiblement différentes : la première exploite les changements temporels dans les images pour *èliminer les parties statiques avant toute interprétation*, tandis que la seconde *interprète tous les éléments des images* au cours d'une segmentation statique visant à extraire les primitives à mettre en correspondance.

La détection du mouvement dans une séquence d'images consiste à distinguer les zones fixes et mobiles d'une scène. Comme le mouvement d'objets induit des différences temporelles entre images, sa détection s'appuie sur l'étude des variations temporelles de la fonction de **luminance** : si le capteur est fixe et que l'éclairement de la scène varie peu, alors nous supposons que **toute variation temporelle de l'intensité est liée au mouvement d'un objet ou à du bruit**.

La détection du mouvement contient les méthodes de différences d'images (approche directe), de corrélation ou de recherche dans l'espace des paramètres.

Les méthodes bas-niveau exploitent la comparaison pixel à pixel, ou petite région à petite région entre deux images consécutives d'une séquence. Elles permettent de déterminer les régions de variations de l'image dans le temps. Elles nécessitent soit une caméra fixe, soit un recalage préalable dans le cas d'un observateur mobile, afin de détecter uniquement les zones de mouvement dans la scène. Elles sont généralement limitées aux mouvements d'objets rigides[2] et au cas de petits déplacements.

[2]pas de déformations de l'objet mobile entre deux images consécutives

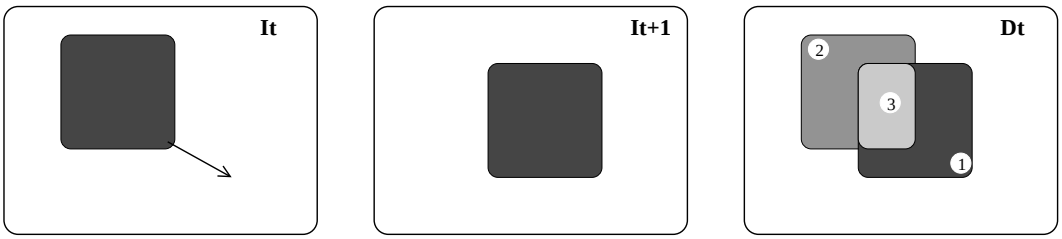



FIG. 3.1 – Illustration schématique de trois régions de mouvement distinctes : (1) objet mobile ;(2) phénomène de fantôme ; (3) phénomène de recouvrement.

Si l'on considère le mouvement d'un objet de luminance homogène non texturé, sur fond immobile observé par une caméra fixe, l'image des différences contient trois régions distinctes, intéressantes à considérer pour l'analyse de la scène :

1. une région constituée de pixels (pixels couverts) appartenant au fond dans l'image $n$, et à l'objet mobile dans l'image $n+1$ ;
2. une région constituée de pixels (pixels découverts) appartenant à l'objet mobile dans l'image $n$, et au fond à l'image $n+1$ (phénomène de fantôme[3]) ;
3. une région constituée de pixels appartenant à l'objet mobile dans les deux images. Ces pixels ont la même intensité dans les deux images, car l'objet a une luminance homogène mais ne provient pas de la même partie de l'objet.

Ainsi, les régions (2) et (3) retracent l'objet dans l'image $n$, les régions (1) et (3) dans l'image $n+1$. Les régions (1) et (2) sont détectés comme des régions en mouvement, le problème est d'extraire la région commune (3) qui ne l'est pas. La Figure 3.1 illustre une représentation de ces trois régions.

Nous ne nous plaçons cependant pas toujours dans des situations où l'on peut si aisément séparer de telles régions. Nous avons besoin de détecteurs disposant d'une certaine robustesse aux bruits, aux rotations, à la variation lumineuse, etc...

Nous présentons dans un premier temps le contexte de travail dans lequel nous nous sommes placés pour mener notre étude.

Nous effectuons ensuite un tour d'horizon des différentes techniques de détection du mouvement en accord avec le contexte de travail précédemment exposé.

Puis, nous détaillons trois techniques que nous avons implantées sur le circuit, deux d'entre elles étant des méthodes originales :

[3] *ghost problem* en anglais [CGP03]

1. une technique de détection basée sur le calcul d'une moyenne temporelle récursive classique ;
2. un détecteur basé sur les outils de morphologie mathématique que nous avons nommé filtre morphologique oublieux ;
3. enfin, un opérateur statistique de détection du mouvement qui est basé sur une analogie avec la modulation $\Sigma - \Delta$ en électronique, le filtre $\Sigma - \Delta$.

Nous verrons enfin quelles sont les améliorations que l'on peut apporter à ces différents algorithmes sur des architectures plus classiques, puis quels autres opérateurs pourraient faire l'objet d'une étude approfondie dans une poursuite de l'étude des détecteurs de mouvement sur un système à base de rétine numérique.

## 3.3 État de l'art de la détection du mouvement

Toutes les méthodes de détection du mouvement ont deux principales contraintes qui sont l'éclairage de la scène et le mouvement du capteur. En effet, la situation est différente selon que le capteur est fixe, qu'il bouge lentement ou qu'il soit mobile avec un mouvement compensé. De même, un éclairement constant, des variations d'éclairement à basse fréquence ou des changements soudains d'intensité lumineuse seront gérés différemment par un système de détection du mouvement. A ces difficultés s'ajoutent le bruit du capteur (bruit d'acquisition et de numérisation) et la gestion des zones homogènes (lorsque la différence de luminance entre deux instants est inférieure à un certain seuil).

La détection du mouvement débouche sur de nombreuses applications, dont la compression et l'indexation vidéo, les applications propres à la robotique (détection d'obstacles, égo-mouvement ou guidage de missile par exemple) ou encore celles liées à la télésurveillance (intrusion, trafic routier ou poursuites de cibles). Ainsi, un grand nombre d'algorithmes de détection du mouvement a déjà été proposé. Nous pouvons les classer en quatre grandes catégories en fonction des types de calculs inter-trame effectués, c'est-à-dire en fonction de la façon dont la différentiation temporelle est menée.

La première catégorie d'algorithmes de détection de mouvement regroupe les méthodes différentielles que nous pouvons subdiviser encore en trois groupes :

1. Le premier est basé sur le calcul d'un **gradient temporel** : une mesure de vraisemblance du mouvement est fournie par le changement instantané calculé entre deux trames consécutives [BL93]. Ces méthodes sont naturellement adaptatives aux changements d'environnements, mais sont aussi dépendantes de la vitesse et de la

taille des objets en mouvement. Ce défaut peut être minimisé en utilisant des combinaisons de filtres spatio-temporelles mais au prix d'une complexité croissante.

2. Le second est basé sur des techniques de **différence au fond**[JN79] [KB90] [TKBM99] [CK03] [Pic04] [SG00] qui utilise une image de référence (le fond), représentant les éléments stationnaires de la scène. Ici la mesure de vraisemblance du mouvement est la différence entre la trame courante et le fond. Ces méthodes sont moins dépendantes de la vitesse et de la taille des objets. Cependant, l'adaptation aux environnements dynamiques est une tâche bien plus ardue, pénalisant la détection des mouvements de faible amplitude (des objets petits, très lents ou très peu contrastés).
3. le troisième groupe regroupe les techniques de d/'etections basées sur des frontières mobiles[rk91] que nous ne détaillerons pas ici malgré le fait qu'elles permettent d'obtenir d'excellent r'esultats, notamment sur des architectures identiques à celle sur laquelle nous avons travaillé.

La seconde catégorie est basée sur le calcul de la vitesse locale apparente (**flot optique**) [BB95] qui est utilisée en entrée de la segmentation spatiale [RD03]. Ces méthodes engendrent des informations riches mais sont généralement plus complexes à calculer et sont aussi plus sensibles à la fiabilité des résultats issus du flot optique. Ainsi un compromis doit-être trouvé entre la régularité du flot optique et la précision de la segmentation.

La troisième catégorie est basée sur la modélisation markovienne particulièrement adaptée au traitement d'image bas-niveau car intégrant l'ensemble des propriétés locales des images. Cette modélisation engendre des calculs qui sont locaux et donc massivement parallélisables ce qui la rend intéressante pour le développement d'algorithmes temps réel.

Plus récemment, des **filtres morphologiques** [FMS97] [Sal02] [ARH00] ont été employés pour analyser des séquences vidéo. En utilisant un élément structurant spatio-temporel, une amplitude de variation locale peut-être calculée comme mesure de vraisemblance du mouvement. Une telle mesure peut être utile pour détecter de petites amplitudes de mouvement, mais comme elle est sensible au bruit spatial, elle est souvent intégrée sur des régions entières en utilisant des opérateurs connexes.

Nous allons maintenant détailler un peu plus les différentes approches avant de présenter celles que nous avons retenues pour l'implémentation sur notre système de vision.

### Différence inter-trames

Nous retrouvons ici les techniques les plus simples comme la différence directe entre deux trames consécutives ou bien encore un "ET" logique entre deux différences consécutives préalablement seuillées. La première étape de détection concerne l'analyse de la dérivée

temporelle de la fonction de luminance qui est, dans le cas de fonctions discrètes, approchée par une différence inter-images. Sachant que dans une séquence d'images obtenue à l'aide d'une cam'era fixe, les éléments statiques de la scène apparaissent d'une image à l'autre aux mêmes endroits, on peur éliminer ces éléments par une simple différence d'images pixel à pixel. Les éléments qui restent après cette opération correspondent aux positions consécutives des objets en mouvement.

**Définition 13 (différence d'image)**

$$\zeta(s) = |\frac{dI(s)}{dt}| = |\Delta_{I_{t,t-1}}(s)| = |I_t(s) - I_{t-1}(s)|$$

Théoriquement, en l'absence de mouvement, $\zeta = 0$ et en présence de mouvement $\zeta \neq 0$ (hormis pour un pixel intérieur à l'objet pour un petit déplacement). Les séquences d'images réelles n'étant jamais exemptes de bruit (il y a toujours au moins le bruit lié au capteur), il est préférable de fixer un seuil qui permet de filtrer l'information pertinente.

Finalement, la détection des changements temporels significatifs est réalisée grâce au seuillage suivant :

$$\Psi_{\theta_{diff}}(s) = \begin{cases} 0 & \text{si} \quad \zeta(s) < \theta_{diff} \\ 1 & \text{sinon} \end{cases}$$

où $\theta_{diff}$ est un seuil prédéfini. Cette opération conduit à une carte binaire qui indique les zones de variations significatives de la luminance d'une image à l'autre.

Afin de réduire le bruit et de renforcer la robustesse des détecteurs de mouvement, différentes méthodes sont utilisées. Une première technique réduit le bruit présent sur la différence d'images en réalisant un filtrage passe-bas[4] préalablement à l'opération de seuillage.

Une autre technique consiste à réaliser une opération de "ET" logique entre les cartes de changements temporels calculées à partir de trois images successives. La carte des observations ainsi constituée est mise en œuvre selon la formule :

$$O_t(s) = min(|\Delta_{I_{t,t-1}}(s)|, |\Delta_{I_{t+1,t}}(s)|)$$

Cette dernière méthode est très rapide mais elle ne donne de bons résultats que dans le cas des mouvements suffisamment importants pour éliminer les régions de recouvrement.

[4]par l'intermédiaire de masques de convolution

**Technique de détection robuste de [Bel96]**

On définit les techniques de différence d'images suivantes : une technique utilisant un seuil $\theta$ de mouvement minimal (de différence de niveau de gris), une image $I(s)$, $O_t^1(s) = |I_t(s) - I_{t-1}(s)|$ et une technique utilisant des différences entre l'image courante et une image de référence $O_t^2(s) = |I_t(s) - I_t^{ref}(s)|$. L'image de référence peut être construite de différente manière (voir partie suivante). La Figure 3.2 présente schématiquement l'algorithme de calcul d'une différence d'image robuste.

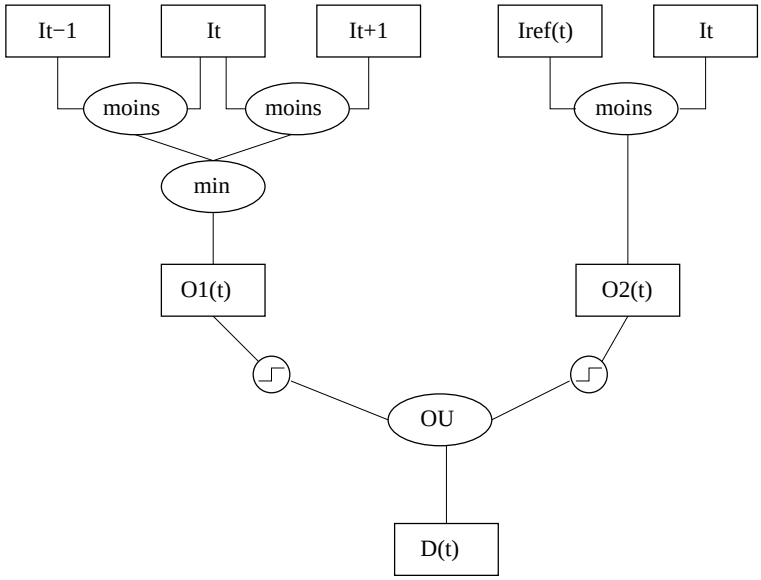



FIG. 3.2 – Différence d'image robuste [Bel96].

Le détecteur robuste est définie par :

– un détecteur à court terme :

$$\Omega_t^1(s) = min(O_t^1(s), O_{t+1}^1(s))$$

– un détecteur à long terme :

$$\Omega_t^2(s) = O_t^2(s)$$

– pour finir la différence d'image robuste est définie par :

$$D_(t) = O^3(t) = max(Seuillage_\theta(\Omega_t^1(s)), Seuillage_\theta(\Omega_t^2(s)))$$

Cette technique a l'avantage d'être rapide mais donne des résultats très bruités. Elle sert donc généralement dans la détection de grosses cibles sans ou avec peu de bruit spatial.

Afin de réduire le bruit issu de la détection et de permettre une bonne segmentation des résultats, une simple différence d'images ne suffit pas. C'est pourquoi on utilise des filtrages pré ou post détection, et en particulier des filtrages morphologiques. Nous abordons très largement le choix de ces filtrages dans la suite de ce chapitre. Notons seulement que le choix d'un filtrage morphologique est judicieux dans le cas d'une segmentation à la suite de la détection. On se retrouve alors dans le cas d'image binaire où la faible complexité et la robustesse des outils morphologiques font que ces filtrages sont appropriés. Cependant, la robustesse du filtrage sera lié au système lui-même et à son domaine d'application.

### 3.3.1 Estimation du fond statique

Le principe des méthodes de différence par rapport à un fond statique ([DH96] [WADP97] [TKBM99] [EHD00] [LH02] [CK03] [CKHL03] [MP04]) est de construire un modèle de la scène indépendant des objets mobiles qui la traversent, de manière à ce que :

1. la différentiation à la base de la détection se fasse entre l'image courante et une image de référence de la scène au lieu de se faire entre images consécutives ;
2. les caractéristiques fines de la scène en terme de bruit temporel soient connues, de manière à avoir un filtrage des fausses alertes spatialement plus adaptatif.

Le calcul du fond statique se fait généralement par une moyenne qui peut prendre différentes formes (moyenne arithmétique, moyenne récursive, max, min, etc...). Pour construire un modèle statique de la scène ; deux approches sont envisageables. La première consiste à accumuler l'information contenue dans la séquence d'images, mettant ainsi progressivement en évidence les parties immobiles de la scène, et effacçant ce qui bouge[Pic04]. un bon effacement demande cependant un grand nombres d'images. Dans la seconde approche, il s'agit de remplacer les objets mobiles dans une image particulière $I_t$ par des éléments du fond cachés dans cette image, mais visibles dans une autre[JN79]. Pour ce faire, on analyse l'évolution de la différence entre l'image $I_t$ et les images consécutives. il est toutefois difficile de fixer un seuil permettant de distinguer les différences dues au mouvement, de celles qui sont provoquées par le bruit ou par des fluctuations de luminosité dans la scène.

**Définition 14 (Moyenne arithmétique)**

$$M_t = \frac{1}{t}(I_t + (t-1)M_{t-1})$$

La moyenne arithmétique n'est envisageable qu'à court terme car le calcul numérique sature dès que la dynamique de $t$ approche celle de $M$. De plus, les calculs sont difficiles à implanter sur un système comme celui de la rétine artificielle. En effet, pour ce faire, il nous faut effectuer pour chaque trame une division par $t$, équivalente sur la rétine à des sommes de décalages[5] de mot binaires.

**Définition 15 (Moyenne récursive)**

$$M_t = \alpha I_t + (1-\alpha)M_{t-1}$$

La moyenne récursive, en revanche, se calcule indépendamment de $t$, en utilisant le paramètre $\alpha$ ($0 < \alpha < 1$), qui est un paramètre d'*oubli*. En effet, lorsque $\alpha$ tend vers 1, la moyenne récursive tend vers l'image courante et lorsque $\alpha$ tend vers 0, la moyenne récursive tend vers la moyenne arithmétique. Numériquement, on est limité par la dynamique de $\frac{1}{\alpha}$ qui ne peut dépasser celle de $M$. Il est à noter que cette expression du calcul de la moyenne correspond au schéma d'implémentation récursif du filtre exponentiel.

Une fois le fond statique déterminé par la moyenne, la différentiation est faite pour la détection par $D_t = |M_t - I_t|$. Un intérêt majeur des techniques de fond statique est leur capacité à détecter des cibles très petites ou qui bougent lentement. En revanche, les filtres récursifs rendent moins précise sur cibles étendues la segmentation des objets mobiles par rapport à des techniques comme celle des modèles markoviens.

De la même façon que l'on calcule des primitives statistiques du premier ordre, la moyenne, nous pouvons calculer des statistiques du second ordre, la variance, l'écart-type.

**Définition 16 (Variance récursive)**

$$V_t = \alpha V_t + (1-\alpha)[M_t - I_t]^2$$

[5]décaler à droite un mot binaire codé sur $n$ bits de $p$ position en introduisant $p$ zéros à gauche revient à diviser le mot par $2^p$

Une amélioration apportée aux méthodes différentielles est l'utilisation de **gaussiennes multiples** et de calculs dans l'histogramme des différences [SG99] [SG00] [CGP03] [Pic04]. Les modèles adaptatifs de distributions gaussiennes multiples sont employés afin de modéliser des fonds complexes (multi-modaux). Chaque pixel est classé en fonction de la Gaussienne (modèle de fond) qui le représente le mieux. Ainsi, ces techniques permettent de suivre des objets en mouvement dans une scène complexe ayant plusieurs modes de mouvement. Les gaussiennes multiples peuvent permettre par exemple de détecter plusieurs objets avec des occlusions et de les différencier de leurs ombres. On peut aussi utiliser des gaussiennes sur chaque canal rouge, vert et bleu mais il devient souvent difficile de les différencier les unes des autres. D'une manière générale le point-clef de ce type de méthodes est la technique employée afin de séparer les gaussiennes.

Le mélange de gaussiennes permet de conserver un historique de la variation d'intensité des pixels. A chaque instant $t$, toute l'information dont nous disposons sur un pixel particulier $\{x_0, y_0\}$ est son historique :

$$\{X_1, \ldots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

où $I$ est une séquence d'images.

L'historique courant de chaque pixel $\{X_1, \ldots, X_t\}$ est modélisée par un mélange de $K$ distribution gaussienne. Cette probabilité d'observation de la valeur courante du pixel est donnée par :

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

où $K$ est le nombre de distributions, $\omega_{i,t}$ est une estimation du poids (quelle portion de donnée est représentative pour cette gaussienne) de la $i^{eme}$ gaussienne dans le mélange à l'instant $t$, $\mu_{i,t}$ est la valeur moyenne de la $i^{eme}$ gaussienne dans le mélange à l'instant $t$, $\Sigma_{i,t}$ est la matrice de covariance de la $i^{eme}$ gaussienne dans le mélange à l'instant $t$ et où $\eta$ est la fonction de densité de la probabilité gaussienne :

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{1}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T\Sigma^{-1}(X_t-\mu_t)}$$

La dynamique de $K$, c'est-à-dire le nombre de modes de mouvement observés, est limitée par la mémoire disponible et la capacité de calcul. En effet, pour trois mouvements distincts, nous aurons trois gaussiennes donc, pour chaque pixel, trois fois plus de données à stocker en mémoire. On comprend vite en regardant les caractéristiques de notre circuit qu'il n'est pour le moment pas raisonnable d'utiliser une telle technique sur un système de vision rétinien.

### 3.3.2 La modélisation markovienne

L'approche markovienne en traitement d'images est très utilisée dans des domaines aussi variés que la restauration d'images bruitées, la segmentation d'objets, la modélisation de textures ou la synthèse d'images texturées, la classification d'images, l'extraction de contours, ou encore la détection du mouvement et l'analyse de scènes dynamiques. La modélisation markovienne, associée aux techniques d'estimation bayésienne, est en réalité une approche statistique qui permet de prendre en compte diverses informations contextuelles. Ces informations généralement issues des relations statistiques existent entre un point et son voisinage et s'expriment sous la forme d'interactions spatiales ou temporelles, formant ainsi un ensemble de connaissances a priori. En considérant le mouvement comme une variation de l'intensité des pixels dans l'image et comme un phénomène aléatoire, nous pouvons modéliser le mouvement comme un événement probabiliste [BL93] [MHC94] [Hen96] [CDLC96].

**Définition 17** *Une suite d'images aléatoire $X$ est une fonction de $\Omega \times \mathbb{Z}^3 \times \mathbb{N}$ dans $\{0, \ldots, 1\}$, où :*
- *$\Omega$ est l'univers probabiliste,*
- *$\mathbb{Z}^3$ est l'espace discret,*
- *$\mathbb{N}$ est le temps discret,*
- *$N \in \mathbb{N}$ est le nombre de niveaux de gris.*

*Pour $s \in \mathbb{Z}^2 \times \mathbb{N}$, $X(s) : \Omega \to \{0, \ldots, 1\}$ est la variable aléatoire au site $s$.*
*Pour $w \in \Omega$, $X(\omega) : \mathbb{N} \to \{0, \ldots, N\}$ est la suite d'images correspondant à l'évènement $\omega$.*
*$\mathbb{S} = \mathbb{Z}^2 \times \mathbb{N}$ est l'ensemble des sites et $\mathbb{V} = \{0, \ldots, N\}$ est l'ensemble des niveaux des gris, et $E = \mathbb{V}^{\mathbb{S}}$ l'ensemble des suites d'images.*

#### Chaîne de Markov

**Définition 18 (chaîne de Markov)** *Une suite $(X_n)_n$ de variables aléatoires à valeurs dans $\mathbb{E}$ est une chaîne de Markov si et seulement si, pour tout $n \in \mathbb{N}$, pour tout $x_0, \ldots, x_n$ de $\mathbb{E}$ tels que $P(X_0 = x_0, \ldots, X_n = x_n) > 0$, et pour tout $x_{n+1}$ de $\mathbb{E}$ :*

1. $P(X_{n+1} = x_{n+1} | X_0 = x_0, \ldots, X_n = x_n) = P(X_{n+1} = x_{n+1} | X_n = x_n)$
   *et* $\forall k > 0$ *et* $\forall n > 0, P(X_{n+k+1} = x | X_{n+k} = x') = P(X_{n+1} = x | X_n = x')$
2. $P(X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n) = q_{i_0} p_{i_0 i_1} p_{i_1 i_2} \ldots p_{i_{n-1} i_n}$
   *où* $\forall i, q_i = P(X_0 = x_i)$ *est la loi de probabilité initiale et*

$$\forall i,j,n \in \mathbb{N}, p_{ij} = P(X_{n+1} = x_j | X_n = x_i)$$
$$\Rightarrow \begin{cases} \forall i, q_i \geq 0 \ et \ \sum_{i=1}^{n} q_j = 1 \\ \forall i,j, p_{ij} \geq 0 \ et \ \sum_{k=1}^{n} p_{ik} = 1 \end{cases}$$

Une chaîne de Markov est caractérisée par la donnée, d'une part $p_{ij}, 1 \leq i,j \leq r$, que l'on appelle les probabilités de transition de l'état $i$ à l'état $j$, et d'autre part des probabilités initiales $q_i, 1 \leq i \leq r$. Les $p_{ij}$ forment la matrice de transition de la forme :

$$p = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & \cdots & \cdots & p_{2n} \\ \vdots & & & \vdots \\ p_{n1} & \cdots & \cdots & p_{nn} \end{pmatrix} \quad \text{telle que } \forall i,j, p_{ij} \geq 0 \text{ et } \sum_{k=1}^{n} p_{ik} = 1$$

Remarque : Soient $q_i^{(n)} = P(X_n = x_j | X_0 = x_i)$, la probabilité (inconditionnelle) que la chaîne de Markov soit dans l'état $i$ à l'instant $n$, et soit $p_{ij}^{(n)} = P(X_n = x_j | X_0 = x_i)$, la probabilité conditionnelle de passer de $x_i$ à $x_j$ en $n$ étapes.

Ainsi, $q_i^{(n)} = \sum_{j=1}^{n} q_j p_{ij}^{(n)}, \forall i = 1, \ldots, n$

1. $\forall k \geq 0, \forall i,j = 1, \ldots, n, p_{ij}^{(n)} = P(X_{n+k} = E_j | X_k = E_i)$
2. $\forall n \geq 0, \forall ij = 1, \ldots, r,$
$$\begin{aligned} p_{ij}^{(1)} &= p_i j \\ p_{ij}^{(n+1)} &= \sum_{k=1}^{r} p_{ik} p_{kj^{(n)}} \end{aligned}$$

**Propriété 4**

$$\begin{aligned} P(X_{n+1} = Ei_{n+1}, \ldots, X_{n+k} = E_{i_{n+k}} | X_0 = E_{i_0}, \ldots, X_n = E_{i_n}) \\ = p_{i_n i_{n+1}} \ldots p_{i_{n+k-1} i_{n+k}} \\ = P(X_1 = E_{i_{n+k}}, \ldots, X_k = E_{i_{n+k}} | X_0 = E_{i_n}) \end{aligned}$$

*[AG92]*

### Application à la détection du mouvement

L'approche markovienne est pertinente dans le cadre de la détection du mouvement car il n'est pas rare qu'une scène abrite des régions (ou des pixels) ou le mouvement est plus souvent observés (plus probable). Moyennant l'hypothèse de caméra fixe et d'éclairement quasi constant de la scène, il existe un lien entre objets mobiles et changements temporels de la fonction de luminance. Cela conduit naturellement à prendre comme observation la valeur absolue de la dérivée temporelle de la fonction de luminance $I(x, y, t)$ qui est approchée numériquement par une différence entre les instants $t$ et $dt$ :

$$Y(x, y, t) = |I(x, y, t) - I(x, y, t - dt)|$$

Par ailleurs, les étiquettes pertinentes dans le cas de la détection sont les suivantes :

$$I(x, y) = \begin{cases} 0 & \text{fixe} \\ 1 & \text{mobile} \end{cases}$$

L'application de la modélisation markovienne à la détection du mouvement se fait en trois étapes :

1. une étape de modélisation (utilisation des **champs de Markov**)
2. une étape de simulation (utilisation des **champs de Gibbs**)
3. une étape d'optimisation (utilisation des algorithmes **ICM**[6] ou **recuit simulé**)

### Champ aléatoire

$X : \Omega \to V^S$ où $S$ est l'ensemble des sites (ou des pixels) d'une image et $V$ est l'ensemble des valeurs des sites.

$\forall w \in \Omega, X_w : S \to V$ est la réalisation d'un champ aléatoire.

$\forall s \in S, X_s : \Omega_s \to V$ est la variable aléatoire du pixel

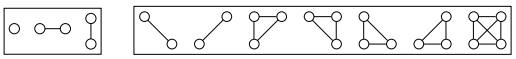
$V = \{0, 1\} = \{\text{fixe}, \text{mobile}\}$ et $S = \mathbb{Z}^3$ ou $S = \mathbb{Z} \times \mathbb{N}$

topologie sur $S$ : relation de dépendance des v.a. $X_s$

[6] *Iterated Conditional Modes*

### Champ de Markov

Un champ de Markov est défini relativement à un voisinage. A ce voisinage sont associées des cliques, définies comme étant des sous-ensembles de sites voisins du site $s$, incluant ce site et tels que deux sites de la clique soient toujours mutuellement voisins.



(1) (2)

FIG. 3.3 – La forme des différents cliques sur $\mathbb{Z}^2$ en 4-connexité (1) et en 8-connexité (2)

**Définition 19** *Le champ aléatoire $X$ est un champ de Markov relativement au système au voisinage de $\mathcal{V}$ si et seulement si pour tout $s \in \mathbb{S}$ et pour tout $x_r \in \mathbb{V}$ :*
$P(X_s = x_s \ / \ X_r = x_r \ , \ r \neq s) = P(X_s = x_s \ / \ X_r = x_r \ , \ r \in \mathcal{V}_s)$

La seconde condition induit une propriété de localité : la valeur d'une étiquette en un site ne dépend pas des étiquettes de tous les autres sites de l'image, mais uniquement des étiquettes des sites voisins.

Le champ des étiquettes est estimé au sens du critère du Maximum A Posteriori (MAP). Il conduit à la recherche de la configuration la plus probable du champ d'étiquettes par maximisation de la probabilité conditionnelle des étiquettes relativement aux observations. Ainsi, pour détecter le mouvement, nous recherchons la réalisation la plus probable d'un champ de Markov dit "caché", à partir d'un champ connu dit "observation".

### Champ de Gibbs

**Définition 20 (Mesure de Gibbs)** *$X : \Omega \rightarrow E^S$ est un champ de Gibbs s'il existe une fonction $U$ (énergie), $U : E^S \rightarrow \mathbb{R}$ telle que :*

$$P(X = x) = \frac{e^{-U(x)}}{Z}$$

*avec $Z = \sum_{x \in V^S} e^{-U(x)}$ représente une constante de normalisation, nommée fonction de partition ou mesure de Gibbs (ou de Boltzmann) d'énergie $U$.*

**Propriété 5 (Théorème de Hammersley-Clifford)** *Soit $X$ un champ aléatoire à valeur dans $\mathbb{E}$ tel que $\forall x \in \mathbb{E}, P(X = x) > 0$. $X$ est un champ de Markov relativement au système de voisinage $\mathcal{V}$ si et seulement si sa distribution $P(X = x)$ est une mesure de Gibbs associée à $\mathcal{V}$.*

$U(x)$ est la fonction d'énergie associée au modèle *a priori* et s'exprime sous la forme :

$$U(x) = \sum_{c \in C} V_c(x)$$

où $C$ désigne l'ensemble des cliques de l'image. Chaque terme $V_c(e)$ est une fonction de potentiel élémentaire associée à une clique $c$ donnée. La Figure 3.4 présente les paramètres intervenant dans le calcul de la fonctionnelle d'énergie dans le cas du 10-voisinage spatio-temporel (8 voisins spatiaux et 2 voisins temporels).

**Définition 21 (Modèle de Potts)** $V_x(s,r) = \begin{cases} -\beta_{sr} \ si \ x(s) = x(r) \\ +\beta_{sr} \ si \ x(s) \neq x(r) \end{cases}$

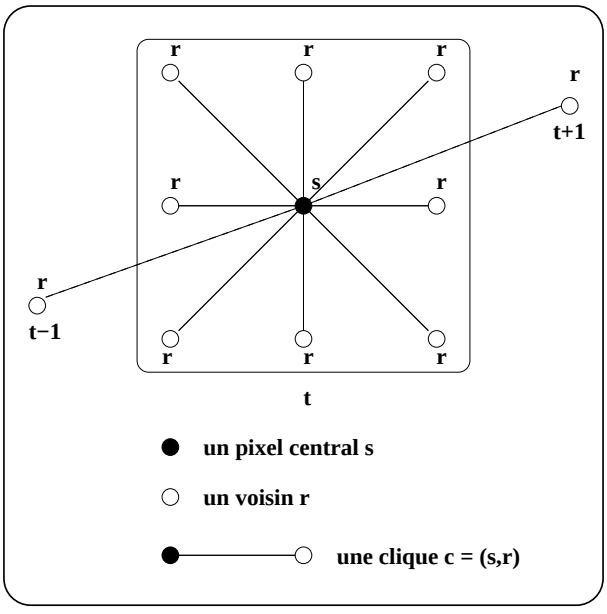



FIG. 3.4 – Les paramètres intervenant dans le calcul de la fonctionnelle d'énergie dans le cas du 10-voisinage spatio-temporel (8 voisins spatiaux et 2 voisins temporels).

Ces potentiels sont définis spécifiquement vis-à-vis du problème à résoudre, la seule contrainte à respecter est que chacun de ces potentiels ne dépende que des sites de la

clique $c = (s, r)$. Ces fonctions de potentiel permettent d'introduire des contraintes sur les solutions recherchées dont celle d'homogénéité spatiale du champ des étiquettes[7].

$$U(x) = U_m(x) + U_a(x, y)$$

Le terme $U_m$ est appelé énergie du modèle. Il exprime une hypothèse de régularité par des potentiels qui mesurent la disparité.

$$U_m(x) = \sum_{s \in S} \sum_{r \in V^S} V_x(s, r) \simeq \sum_s \sum_r \pm\beta_{sr}$$

Le terme $U_a$ est appelé énergie attachée aux données ou énergie d'adéquation. Il mesure un lien significatif entre le résultat de la segmentation des données du problème. Son rôle consiste à éviter que le terme de régularisation représenté par l'énergie du modèle n'éloigne trop le résultat de l'initialisation.

$$U_a(x, y) = \frac{1}{2\sigma^2} \sum_{s \in S} (y(s) - \Psi(s))) \text{ avec } \Psi(s) = \begin{cases} 0 \text{ si } x(s) = 0 \text{ fixe} \\ \beta_{sr} \text{ si } x(s) = 1 \text{ mouvement} \end{cases}$$

On suppose $P(X = x) = \frac{e^{-U_m(s)}}{Z_s}$ et $P(Y = y|X = x) = \frac{e^{-U_\alpha(x,y)}}{Z_r}$ (modèle de bruit liant $X$ et $Y$).

alors $Argmin_x U(x) = argmax_x P(X = x)P(Y = y|X = x) = argmax_x P(X = x|Y = x)$ critère Bayesien du Maximum a priori

**Conclusion**

La minimisation de la fonction énergie est un problème non trivial car cette fonction est a priori non convexe. Les principales techniques sont les algorithmes de Recuit-simulé qui offrent de bons résultats mais dont le coût de calcul est élevé et les algorithmes ICM qui sont des algorithmes de relaxation déterministe. Ces algorithmes sont sous-optimaux, puisqu'ils n'autorisent des changements d'étiquettes que si ceux-ci engendrent une diminution d'énergie. Ils ne garantissent donc pas de trouver le minimum global et sont donc très dépendants de l'initialisation [Dum96] [BL93]. De plus, les modèles d'étiquetage existants sont fondés sur certaines hypothèses restrictives, par exemple : le déplacement de l'objet mobile entre images voisines doit être supérieur à sa taille ou,

[7]les configurations favorisées sont celles où deux étiquettes voisines sont identiques

dans le contraire, cet objet ne doit pas être uniforme. De tels algorithmes sont complexe à intégrer dans un circuit comme celui de la rétine numérique. Nous verrons dans la suite du manuscrit comment toutefois envisager une implantation, notamment avec l'aide de fonctions asynchrones dont on disposerait dans les futurs modèles de rétines[Gie05].

### 3.3.3 Les techniques morphologiques

L'utilisation de la morphologie mathématique en détection et en estimation du mouvement est relativement récente. Elle a donné lieu à des développements intéressants pour les systèmes de détection, de poursuite et de reconnaissance d'objets, dans la mesure où elle intègre naturellement des notions de plus haut niveau telle que la taille ou la forme des objets dans les traitements élémentaires. Les techniques morphologiques rencontrées dans la littérature peuvent être classées dans deux catégories majeures :

1. les techniques qui font coopérer segmentation statique de type ligne de partage des eaux avec une analyse du mouvement [Vin93][BT00][Sal02]. C'est le cas pour une segmentation topologique de l'image où la topologie est contrainte par des **plots** représentant la position estimée ou prédite des objets.
2. les techniques opérant par simplification hiérarchique de l'image en zones plates et calculant les attributs du mouvement sur ces zones plates et non sur les pixels [PVT93][IDE97][ARH00][WL97].

Ces algorithmes nous semblent particulièrement intéressants dans la mesure où ils permettent d'envisager une collaboration plus fine et mieux rebouclée entre les différents étages du système en appliquant la détection directement au niveau "de l'objet". Néanmoins, ils ne pourront être véritablement efficaces que sur une rétine pourvue d'opérations associatives asynchrones[8], car le calcul au niveau région suppose une algorithmique très

[8]L'asynchronisme désigne le caractère de ce qui ne se passe pas à la même vitesse, que ce soit dans le temps ou dans la vitesse proprement dite, par opposition à un phénomène synchrone.

- En informatique, des exemples de phénomènes asynchrones sont la communication par courriel ou la connexion ADSL où la vitesse de l'envoi est indépendante de la vitesse de la réception. La mise en œuvre informatique utilise des files d'attente pour traiter des données qui arrivent de façon asynchrone. Réciproquement pour synchoniser des processus asynchrones on utilise un mécanisme conceptuel appelé sémaphore (en informatique) dû à Edsger Dijkstra. Du point de vue du programmeur, une méthode est asynchrone si elle se termine sans attendre de réponse.
- En télécommunication, on utilise le mode asynchrone pour palier le problème de calage d'horloge entre les données, celle-ci sont émises n'importe quand, puis on resynchronise les horloges à chaque émission.
- En éléctronique, un compteur asynchrone est un compteur qui comporte l'inconvénient de ne pas pouvoir faire sortir l'état du compteur sur toute ses broches en même temps (mise à jour en cascade). Avec un compteur asynchrone ces 8 broches ne seront pas instantanément mise à jour en même temps.

irrégulière peu adaptée à la maille SIMD synchrone.

L'utilisation d'éléments structurants temporels ou spatio-temporels permet d'aborder différemment la différentiation trame à trame. Elle permet à la fois de calculer des filtres spatio-temporels intéressants, et d'intégrer des changements temporels par accumulation (lorsque l'élément structurant est allongé dans l'axe temporel). Elle permet d'autre part de discriminer un déplacement donné en orientant l'élément structurant dans la direction correspondante.

L'espace-temps discret est assimilé à $\mathbb{Z}^3$ avec deux dimensions spatiales et une dimension temporelle, un élément structurant spatio-temporel $B$ est défini comme un voisinage de l'origine dans $\mathbb{Z}^3$[ARH00].

Considérons une séquence d'image $I_t(x)$ où $t$ est un index de temps et $x$ un index d'espace (bidimensionel), les filtres morphologiques temporels sont définis en utilisant un élément structurant temporel[9] $\tau = [t_1, t_2]$ avec $t_1$ et $t_2$ deux instants tels que $t_1 \leq t_2$.

**Définition 22 (Erosion temporelle)** *L'érosion temporelle en niveau de gris est définie par :*

$$\varepsilon_\tau(I_t)(x) = \min_{z \in \tau}\{I_{t+z}(x)\}$$

**Définition 23 (Dilatation temporelle)** *La dilatation temporelle en niveau de gris est définie par :*

$$\delta_\tau(I_t)(x) = \max_{z \in \tau}\{I_{t+z}(x)\}$$

**Définition 24 (Gradient morphologique temporel)** *Le gradient (morphologique) temporel $\gamma$ est alors défini par :*

$$\gamma_\tau(I_t) = \delta_\tau(I_t) - \varepsilon_\tau(I_t)$$

*(extension des formules spatiales)*

où $\tau$ représente un intervalle temporel d'interaction, qui peut-être causal (i.e. $[-3, 0]$), anti-causal (i.e.. $[0, +5]$) ou les deux (i.e. $[-1, +1]$). De tels gradients temporels correspondent à une amplitude de variation.

[9] un interval de temps

De nombreuses autres méthodes de détection du mouvement ont été élaborées qui utilisent des outils de morphologie mathématique, soit pour obtenir une meilleure segmentation des résultats soit pour détecter des configurations particulières correspondant à des objets. De manière générale, les autres opérateurs utilisant la morphologie mathématique sont moins utilisés car ils n'offrent pas de résultats directs comparables aux opérateurs classiques. Les opérateurs géodésiques tiennent une place importante dans ces techniques car ils offrent la possibilité de reconstruire un objet mobile détecté dans la scène originale [FMS97].

## 3.4 La moyenne récursive classique

### 3.4.1 Introduction

Comme nous l'avons vu précédemment, les méthodes d'estimation du fond conviennent bien aux objectifs et aux contraintes de notre système. Nous avons donc naturellement commencé par l'implantation d'une technique basée sur un opérateur facilement adaptable sur le circuit de la rétine, en l'occurrence une moyenne récursive. Cette première étude nous a servi à la fois de base algorithmique sur laquelle nous avons pu tester et valider bon nombre de fonctions élémentaires. Elle a été également une base technique qui nous a permis de comparer les performances des différents algorithmes entre eux en terme de qualité de détection mais aussi en terme de cadence de traitement des images.

Dans le cas d'un capteur fixe et d'une détection à très long terme, cette technique est particulièrement intéressante, car les scènes que nous devons considérer sont très variables, en terme de textures de fond, d'horizons, et d'éléments perturbants (nuages, arbres, cours d'eau, ...) dans l'espace, mais aussi dans le temps (éclairement en fonction du jour et de la nuit, position du soleil, force du vent, ...).

### 3.4.2 Principe

**Définition 25 (Moyenne récursive)** *Soit $I_t$, l'image acquise à l'instant $t$, $M_t$ le fond courant et $\alpha \in [0,1]$ une constante. Le fond est calculé récursivement à l'aide de la formule suivante :*

$$\begin{aligned} M_t &= \alpha I_t + (1-\alpha) M_{t-1} \\ &= \alpha (I_t - M_{t-1}) + M_{t-1} \end{aligned}$$

Le calcul est particulièrement simple lorsque $\alpha$ est une puissance négative de 2, et correspond alors à un simple décalage des bits du mot à gauche[10]. Lorsque $\alpha$ vaut 1, $M_t = I_t$, le fond est l'image courante. Lorsque $\alpha$ vaut 0, $M_t = M_{t-1} = M_0$, le fond reste égale à sa valeur initiale. La technique de fond récursif revient dans ce cas à une technique par différence avec une image de référence qui ne serait jamais remise à jour. En faisant varier $\alpha$ au cours du temps, nous pouvons donc envisager de changer la pondération de la moyenne et celle de l'image courante afin de s'adapter aux conditions de la scène.

Les deux formules présentées ci-dessus (Définition 25) sont mathématiquement identiques, cependant nous pouvons remarquer que dans la seconde nous n'effectuons qu'une seule multiplication par $\alpha$. Or, lorsque nous réalisons plusieurs multiplications par $\alpha$, nous multiplions les erreurs d'arrondis. C'est pour cela que dans le cadre du portage sur la rétine, nous avons privilégié la seconde formule.

La Table 3.1 présente l'algorithme utilisé pour la détection de mouvement utilisant une moyenne récursive. Ici, le résultat est issu d'un simple seuillage global de la valeur absolue de la différence entre la moyenne calculée et l'image acquise. Plusieurs raffinements prenant la forme de filtrage spatial sont ensuite présentés en fin de partie afin de rendre la segmentation des objets détectés plus précise.

La Figure 3.19(1) montre les résultats du calcul de fond récursif pour un pixel particulier correspondant à une zone de passage d'un objet en mouvement. On peut voir sur cette figure que l'opération de moyenne récursive $M_t$ correspond à un filtrage de la scène temporelle $I_t$. Plus précisément, il s'agit de l'implantation récursive causale du filtre exponentiel dans le domaine temporel.

| | |
|---|---|
| **Initialisation** | |
| Pour chaque pixel $x$ : | $M_0(x) = I_0(x)$ |
| **Pour chaque trame t** | |
| Pour chaque pixel $x$ : | $M_t(x) = \alpha I_t(x) + (1-\alpha)M_{t-1}(x)$ |
| **Pour chaque trame t** | |
| Pour chaque pixel $x$ : | $D_t(x) = \delta_t(\lvert M_t(x) - I_t(x)\rvert)$ |

TAB. 3.1 – Calcul de la moyenne récursive, $\delta_t$ est le seuillage de $M_t$. On utilise dans un premier temps un seuillage global.

Afin de rendre cette technique plus robuste, nous l'avons enrichie de post-filtrages spatiaux et de techniques d'adaptation au seuillage spatio-temporel. Une fois la détection déterminée, nous pouvons envisager de calculer la moyenne des niveaux de gris des carrés

[10]Par exemple, $[1001]_2 \times 2^{-4} = [1001]_2\ 4 = [0010]_2$

ou des valeurs absolues de $D_t$ pour obtenir les valeurs de variance fournissant alors une **cartographie du bruit temporel**. Cette image renseignant sur les zones de l'image où se situe plus probablement un mouvement perturbateur a été utilisé dans une première tentative d'amélioration de l'algorithme. En effet, nous utilisons l'information fournit par cette carte des bruits nous a permis de mettre en place un système de rebouclage où l'on met à jour certaines valeurs (moyenne récursive) que si le mouvement est supérieur à la valeur issue de la carte des bruits. Les améliorations proposées pour les algorithmes et notamment les prinipes de rebouclage sont plus amplement abordés dans la suite du chapitre.

La Figure 3.5 nous montre le résultat de la détection à base de moyenne récursive sur une séquence de trafic urbain calculée sur un tampon de longueur 8. Nous remarquons dans l'image de la moyenne $M_t$ et surtout de la différence $\Delta_t$, la trace laissée par le déplacement de l'objet au cours de son déplacement.

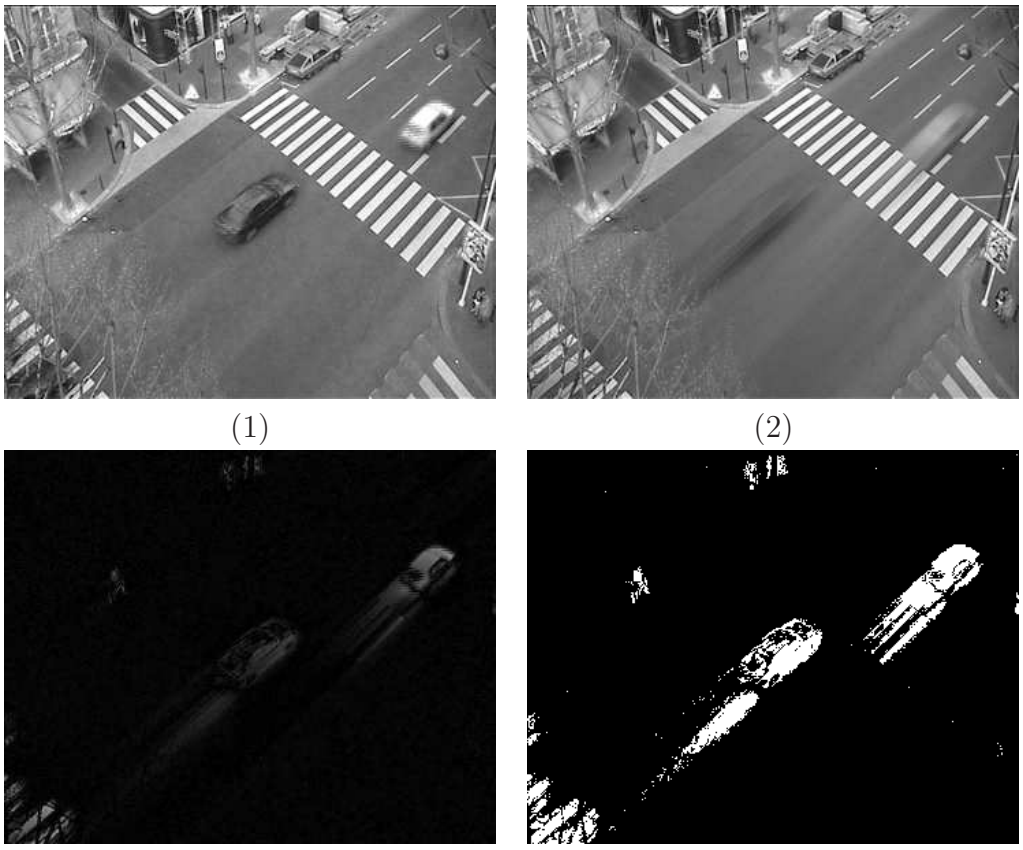
### 3.4.3 Implémentation rétinienne

Une des dificultés du passage de l'algorithmie à l'implémentation sur le circuit rétinien sont les problèmes techniques liés aux calculs entiers. En effet, on ne peut raisonnablement pas envisager d'effectuer des calculs précis en flottant puisque nous ne disposons que d'une quantité llimitée de points mémoires par pixels.

Afin d'obtenir une implantation rapide, nous limitons comme nous l'avons souligné la division à un décalage en prenant $\alpha = 2^{-k}$. Pour une image acquise en $n$ bits, l'intervalle de valeurs pour lesquelles le calcul ait un sens est $\alpha \in [2^{-(n-1)}, 1]$. La multiplication par $\alpha$ revient alors à effectuer un décalage à droite de $k$ bits, $k \in \{0, n-1\}$.

Notons que grâce au bit de signe, lorsque nous ajoutons un nombre positif inférieur à $2^k$, l'effet est nul alors que lorsque nous ajoutons un nombre négatif quelconque, nous décrémentons toujours au moins de 1[11]. Nous rappelons que nous utilisons les opérations élémentaires présentées dans le chapitre précédent (Chapitre 2 Partie 2.3). Nous menons donc le calcul une trame sur deux en ajoutant $\alpha * (X_t - M_t)$ et une trame sur deux en soustrayant $\alpha * (M_t - X_t)$ (voir Table 3.2). Ce calcul est mathématiquement identique mais permet de tenir compte du problème lié aux arrondis. Le fait d'intervertir le sens de la différence une trame sur deux permet statistiquement de pouvoir ajuster au mieux le calcul du fond. En effet, à chaque instant $t$, la probabilité que l'image acquise $X_t$ soit

[11]Pour $M_t < I_t$, on a $M_t - I_t < 0 \equiv [1.....]_2$ donc $\forall \alpha \in [0,1]$, on a $\alpha * (M_t - I_t) > 0$. Si $M_t \neq I_t$ alors $M_{t+1} = M_t - \alpha * (M_t - I_t) \neq M_t$



(1) (2)

(3) (4)

FIG. 3.5 – Résultats de la détection par moyenne récursive $\alpha$ pour une séquence de trafic urbain. (1) $I_t$ (2) $M_t$ (3) $\Delta_t$ (4) $D_t$ ($\alpha = \frac{1}{8}$).
On remarque la détection des piétons en haut de l'image et la trainée laissée par les voitures plus rapides.

supérieure à la moyenne récursive $M_t$ est égale à celle que $X_t$ soit inférieure à $M_t$. Cet algorithme, présenté Table 3.2, rend compte des problèmes typiques que l'on a rencontré au cours de l'adaptation d'algorithmes sur la rétine et aux "erreures" constatées lors des premiers tests. Ce style de techniques, s'apparentant plus parfois à des recettes de cuisines qu'à des développements informatiques classiques, représente en effet le plus gros du travail fournit pendant la thèse.

```
Pour chaque trame t {
  si ((t mod 2)=0) alors{
    X = X - M;
    M = M + α* X;
  }sinon{
    X = M - X;
    M = M - α* X;
  }
}
```

TAB. 3.2 – Algorithme de calcul de la moyenne récursive.

Enfin, afin de décider quels sont les pixels en mouvement, un seuillage global de la valeur absolue de la différence entre $X_t$ et $M_t$ doit être effectuée. L'algorithme de la Table 3.3 nous montre comment effectuer le seuillage de l'image $X_t$ (codée sur la rétine) par rapport à un seuil global $Th$ (codé sur le cortex).

```
Initialisation
T = 1
Pour chaque bit i de X_i
  si (Th[i] == 0)
    alors T = T ∨ X_i ;
    sinon T = T ∧ X_i ;
```

TAB. 3.3 – Algorithme de calcul du seuillage de $X_i$ par $Th$. Avec $T$ l'image binaire représentant le résultat seuillé ($T = 1$ si $X \geq Th$).

### 3.4.4 Raffinements proposés

Comme nous l'avons vu dans les résultats de simulation de la Figure 3.5, la détection brute comporte un certain nombre de défauts dûs aux phénomènes de fantômes et à la

présence de points isolés. Nous verrons que les points isolés issus de la détection peuvent être considérés raisonnablement comme du bruit temporel (voir Chapitre 2 Section 2.3.4 pour plus de détails sur l'implémentation de ce calcul dans la rétine).

Nous avons enrichi notre algorithme de deux modules nous permettant d'obtenir une détection de meilleure qualité puis ensuite d'utiliser un filtrage spatial. Tout d'abord, au lieu de fixer un seuil de manière définitive pour chaque image de la séquence dans l'étape finale de la détection, nous utilisons un seuillage global automatique $S$ basé sur le comptage des points isolés $N$. En effet, à chaque instant $t$, si $N$ augmente alors $S$ augmente.

Enfin, nous avons mis en place une stratégie de rebouclage qui consiste à pondérer la mise à jour de la moyenne $M_t$ en chaque pixel par deux paramètres $\alpha_1$ et $\alpha_2$ ($\alpha_2 << \alpha_1$) en fonction du résultat courant de la détection $D_t$. Cela nécessite un réordonnancement complet de l'algorithme en intégrant la mise à jour suivante :

$$M_t = M_{t-1} + \alpha_1(I_t - M_{t-1}) \times D_t + \alpha_2(I_t - M_{t-1}) \times \overline{D_t}$$

| | |
|---|---|
| **Initialisation** | |
| Pour chaque pixel $x$ : | $M_0(x) = I_0(x)$ |
| **Pour chaque trame t** | |
| Pour chaque pixel $x$ : | $\Delta_t(x) = M_t(x) - I_t(x)$ |
| | $D_t(x) = \delta_t(\lvert\Delta_t(x)\rvert)$ |
| | $M_{t+1}(x) = M_t(x) + \alpha_1(\Delta_t(x)) \times D_t(x) + \alpha_2(\Delta_t(x)) \times \overline{D_t(x)}$ |

TAB. 3.4 – Algorithme du calcul de la moyenne récursive réordonnancé avec rebouclage.

La Table 3.4 présente l'algorithme de la détection du mouvement utilisant la moyenne récursive enrichie de la stratégie de rebouclage que nous avons présentée. En d'autres termes, la moyenne $M_t$ pour le pixel $x$, névolue pas lorsque à l'instant $t$, résultat de la détection est nul sauf si la différence $\Delta_t$ entre le signal et la moyenne est suffisamment grande (de l'orde de $\frac{1}{\alpha_2}$). Cette modification nécessite un réordonnancement complet de l'algorithme (on effectue la détection avant l'éventuellement mise à jour de la moyenne) mais permet de rendre la détection plus adaptatif à l'environnement.

### 3.4.5 Résultats

La Figure 3.6 présente les résultats de la détection par moyenne récursive sur le banc algorithmique. La procédure de test employée pour générer les résultats présentés est la

suivante : le circuit rétinien est couché sur le bureau. Après avoir laissé passer un moment correspondant au temps nécessaire pour que l'algorithme converge (quasi-instantanné en fonction des conditions initiales de luminosité), nous passons notre main plus ou moins rapidement devant la lentille de la rétine.

Par rapport aux résultats présentés précedemment, le filtrage permet de minimiser les phénomènes de fantômes, sans pour autant les éliminer totalement. La technique de rebouclage nous a permis d'être plus réactif aux changements notamment d'illuminations de manière satisfaisante aux vu de la taille mémoire utilisée et du temps de calcul nécessaire (seulement 156 instructions et 23 points mémoires utilisés pour un traitement en 8 bits permettant de traiter une trame en moins de 0,4ms). Cet algorithme nous laisse en effet une marge de manoeuvre appréciable permettant d'envisager l'ajout ultérieure de fonctions d'estimation du mouvement, par exemple. Des mouvements perturbants saccadés comme des bougés seront par contre plus pénalisant quant à la précision (segmentation) des objets mobiles.

### 3.4.6 Conclusion

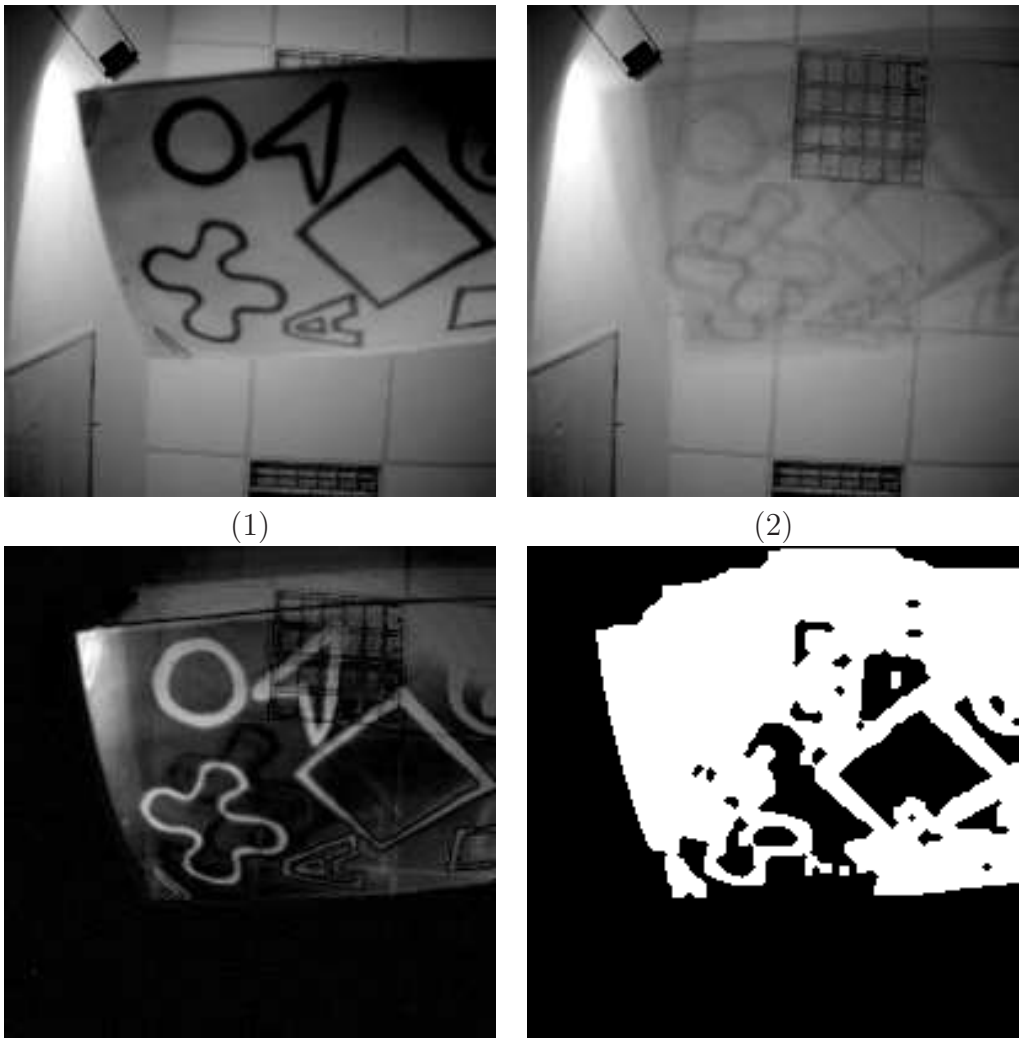
L'algorithme est peu coûteux en mémoire de données mais est moins adaptatif et moins précis en terme de localisation.

Nous retiendrons de ces techniques deux caractéristiques importantes :

1. la notion de **filtre d'accumulation** qui permet de s'adapter à des mouvements très divers ;
2. la notion de **cartographie du bruit temporel** qui permet une adaptation fine et dynamique aux caractéristiques spatiales de la scène.

Divers essais ont été menés afin d'exploiter au mieux la cartographie construite le plus souvent comme une moyenne récursive des détections. Les résultats les plus exploitables sont l'utilisation de cette cartes afin d'éliminer des mouvements survenant dans une zone de l'image où les mouvements sont peu problables nous permettant d'éviter un certains nombre de fausses détections. L'autre principale utilisation a été faite lors du mécanisme de rebouclage où l'on utilise cette carte comme critère de décision pour la mise à jour de la moyenne. Dans tous les cas, le surcoût lié au calcul de la carte des bruits ne se justifie pas quand à l'améliroration des résultats, c'est pourquoi nous ne l'avons que peut utilisé.

Du fait de l'utilisation de filtres d'accumulation récursifs, les phénomènes de rémanence ou de fantômes sont très présents. En effet, un objet mobile laisse au cours de son mouvement une trace sur le fond plus ou moins importante selon sa vitesse et de son contraste



(1) (2)

(3) (4)

FIG. 3.6 – Résultat de la détection par moyenne récursive sur le banc algorithmique : (1) image non traitée $I_t$ ; (2) fond estimé $M_t$ ; (3) différence au fond $\Delta_t$ et (4) détection filtrée $D_t$.

par rapport au fond[12]. Pour remédier à ce problème qui peut être très gênant car pouvant occasionner de fausses détections ou au contraire empêcher le système de détecter un objet mobile, nous devons mettre en place un filtrage spatial. Ce filtrage nous permet en outre d'obtenir une meilleure segmentation des objets détectés. Nous le présentons en détail dans la section suivante qui concerne la détection à l'aide des opérateurs de morphologie oublieuse temporelle.

## 3.5 La morphologie oublieuse temporelle

### 3.5.1 Introduction

La morphologie mathématique étant fondée sur des opérations ensemblistes donc booléennes, elle se prête bien à une implantation sur rétine programmable. Il est donc naturel que nous examinions ces techniques avec une attention particulière. La notion de fond statique peut être exploitée avec profit dans le cadre morphologique des filtres minimum et maximum.

Nous avons naturellement tenté d'étendre certaines notions de mathématique morphologique dans le domaine temporelle. Plus précisemment, nous avons essayer d'utiliser les tampons morphologiques combinés avec les opérations de dilatations et d'érosions morphologiques. On obtient alors, comme nous allons le montrer, un opérateur accumulant pendant une fenêtre temporelle les r/'esultats de maximum et minimum pondérés. C'est parce que l'on travaille avec des fenêtres glissantes que l'on a introduit le terme "oublieux", insistant sur le fait que le mouvement est peu à peu oublié afin de rendre compte de la scène actuelle.

En effet, une technique pouvant être utilisée pour la détection est l'application de filtres minimum et maximum sur une fenêtre temporelle de taille limitée (dite **fenêtre glissante**). Si nous utilisons des fenêtres glissantes, nous perdons tout le bénéfice de l'associativité du min/max. En étant obligé de conserver toutes les trames en mémoire, le coût en capacité mémoire est rapidement prohibitif. Si nous n'utilisons pas de fenêtre glissante, nous évitons ce phénomène, mais un autre problème survient : l'information de détection est disponible de manière intermittente (la différence devient nulle à chaque remise à zéro, puis elle augmente progressivement).

[12]ceci est moins vrai avec les raffinements proposés dont l'adjonction d'une technique de rebouclage, voir 3.7

Une première technique envisagée est alors d'utiliser deux fenêtres non glissantes (dites **tampons morphologiques**), en déphasage, de façon à avoir toujours une information sur le mouvement disponible, et d'obtenir la valeur de détection en moyennant les différences. Cette technique nous permet en outre de filtrer le bruit (qui est maximisé par les tampons morphologiques). Un autre avantage de cette technique est la possibilité de détecter plusieurs types de mouvements très différents, en utilisant des tampons morphologiques de différentes amplitudes.

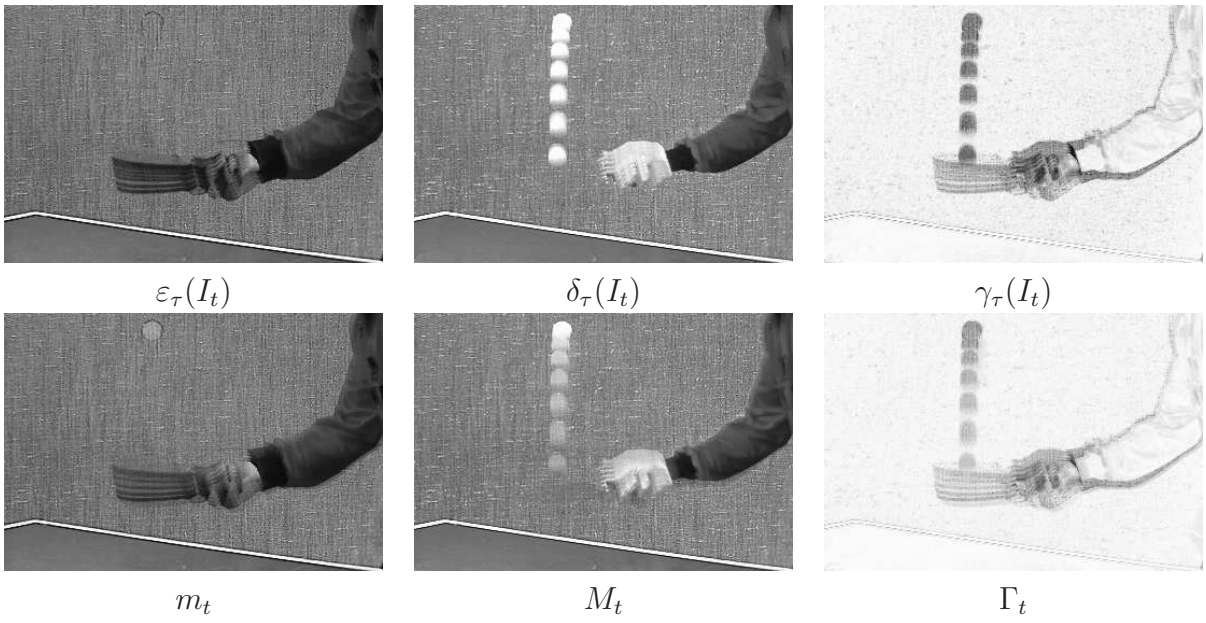



FIG. 3.7 – Application des opérateurs morphologiques oublieux (en bas), comparés aux opérateurs morphologiques classiques (en haut), calculés à la trame $t = 19$ de la séquence "Tennis" (Berkeley).
$\varepsilon_\tau(I_t)$, $\delta_\tau(I_t)$ et $\gamma_\tau(I_t)$ représente respectivement l'érosion, la dilatation et le gradient morphologiques classiques alors que $m_t$, $M_t$ et $\Gamma_t$ représentent respectivement l'érosion, la dilatation et le gradient morphologiques oublieux.
Pour comparaison, les opérateurs morphologiques classiques utilisent pour élément structurant un segment dans la dimension temporelle $\tau = [-8, 0]$ ; et les opérateurs morphologiques oublieux utilisent le paramètre d'oubli $\alpha = 1/9$. Les gradients sont affichés en mode vidéo inversé.
La balle est mieux sementée et sa position actuelle mieux définie par les opérateurs de dilatation et dérosion oublieuses que les opérateurs classiques . De manière générale, le mouvement est mieux détecté (surtout visible dans l'image du gradient morphologique oublieux temporel) alors que les zones homogénes sans bruit mieux filtées.

Une seconde technique envisagée qui s'est révélée particulièrement attrayante est l'uti-

lisation de filtres hybrides maximum et linéaire, et hybrides minimum et linéaire que nous avons nommés **filtres morphologiques oublieux**. Ils combinent en effet le calcul des minimum et maximum avec un terme d'oubli $\alpha$ à l'instar des moyennes récursives (voir Définition 25). La Figure 3.7 illustre le comportement des opérateurs morphologiques oublieux comparés aux opérateurs morphologiques temporels classiques.

### 3.5.2 L'algorithme

L'intérêt de ce filtre hybride est de permettre une accumulation riche comme celle du fond morphologique[13], qui permet d'observer les fortes amplitudes de variation dans une scène, tout en effaçant progressivement les variations les plus anciennes. D'une part, l'ajout d'une partie linéaire dans le filtre permet de mieux filtrer les bruits, tremblements et autres valeurs aberrantes. D'autre part, il cumule l'associativité des opérateurs minimum et maximum et la récursivité, ce qui le rend très peu coûteux en occupation mémoire.

L'utilisation des opérateurs morphologiques classiques souffrent en effet de deux inconvénients majeurs :

1. Ils impliquent l'utilisation d'un buffer d'une taille correspondante au diamètre de l'élément structurant temporel, ce qui peut-être très pénalisant d'un point de vue de l'occupation de l'espace mémoire ;
2. Ils sont très sensibles aux larges variations brusques (comme un bruit impulsionnel ou une oscillation du capteur).

Afin de prendre en compte ces deux problèmes, nous utilisons des filtres hybrides, qui peuvent être vus comme une estimation récursive de la valeur des érosions et dilatations temporelles. En utilisant un paramètre $\alpha$, qui est un réel compris entre 0 et 1, la dilatation temporelle oublieuse $M_t$ (resp. érosion $m_t$) est définie comme le montre la Table 3.5. Comme dans le calcul de la moyenne récursive classique définie par $A_t(x) = \alpha I_t(x) + (1-\alpha)A_{t-1}(x)$, $1/\alpha$ a la dimension du temps. La sémantique de $M_t(x)$ (resp. $m_t(x)$) est alors la valeur maximum (estimée) (resp. minimum) observée au pixel $x$ pendant les $1/\alpha$ dernières images de la séquence. Donc pour $\alpha$ tendant vers 1, $M_t$ (resp. $m_t$) tend vers $I_t$, et pour $\alpha$ tendant vers zéro, $M_t$ (resp. $m_t$) tend vers la valeur maximale (resp. minimale) observée durant la séquence entière.

$\Gamma_t$, le gradient morphologique temporel oublieux, est utilisé par la suite comme un filtre différentiel grâce aux propriétés suivantes :

[13]le fond morphologique est défini par un calcul d'un fond à l'aide des outils de morphologie mathématique, comme par exemple la différence entre les érosions et les dilatations des images ($M_t = E_t - D_t$ avec $E_t = E_{t-1} ET I_t$ et $D_t = D_{t-1} OU I_t$)

| | |
|---|---|
| **Initialisation** | |
| Pour chaque pixel $x$ : | $M_0(x) = m_0(x) = I_0(x)$ |
| **Pour chaque trame t** | |
| Pour chaque pixel $x$ : | $M_t(x) = \alpha\, I_t(x) + (1-\alpha)\, max\{I_t(x), M_{t-1}(x)\}$ |
| | $m_t(x) = \alpha\, I_t(x) + (1-\alpha)\, min\{I_t(x), m_{t-1}(x)\}$ |
| **Pour chaque trame t** | |
| Pour chaque pixel $x$ : | $\Gamma_t(x) = M_t(x) - m_t(x)$ |

TAB. 3.5 – Les opérateurs de filtre morphologique temporel oublieux : $M_t$ est la dilatation oublieuse, $m_t$ l'érosion oublieuse et $\Gamma_t$ le gradient morphologique oublieux.

1. Il a la dimension d'une amplitude de variation temporelle, il est donc capable d'intégrer le mouvement sur une longue période de temps en fonction de $1/\alpha$, et ainsi de détecter de petits mouvements ou des objets lents ;
2. Il est moins sensible aux bruits impulsionnels que le tampon morphologique à cause de son terme linéaire induisant une diminution exponentielle des poids attachés aux valeurs passées ;
3. Il nécessite uniquement l'utilisation d'un buffer de taille 2 (images) pour calculer l'érosion et la dilatation oublieuse.

Le gradient morphologique oublieux représente alors une mesure du mouvement apparent qui peut-être utilisé dans un outil plus complexe de détection du mouvement. Ce filtre rend possible une bonne détection pour les mouvements pour lesquels l'amplitude est sous la discrétisation spatio-temporelle. Cependant, le terme oublieux $\alpha$ a besoin d'être adapté à la vélocité de l'objet en mouvement. Comme il y a plusieurs objets de différentes tailles et vitesses dans la scène, il est nécessaire d'ajuster localement la valeur de $\alpha$ aux observations.

### 3.5.3 Implémentation rétinienne

Afin de mener le calcul du gradient morphologique oublieux sur la rétine numérique, nous reprenons le principe des filtres précédents. En effet, le calcul des érosions et dilatations oublieuses revient à calculer deux moyennes récursives en utilisant en plus un outil de comparaison. La constante $\alpha$ a la même dimension que dans le calcul de la moyenne récursive. De plus, ici nous connaissons le signe des différences entre l'image acquise $I_t$ et les minima $m_t$ et maxima $M_t$. En effet, $I_t - M_t$ et $m_t - I_t$ sont négatifs et nous pouvons donc effectuer les opérations sans problèmes d'arrondis comme le montre la Table 3.6 (Voir Section 3.4.3).

**Pour chaque trame t {**
$M = \mathrm{MAX}(I, M)$ ;
$m = \min(I, m)$ ;
$Y = m - I$ ;
$X = I - M$ ;
$m = m - \alpha * Y$ ;
$M = M + \alpha * X$ ;
$G = M - m$ ;
$D = \delta(G_t)$ ;
**}**

TAB. 3.6 – Algorithme de la détection du mouvement à base d'opérateurs morphologiques oublieux.

La comparaison entre l'image acquise $I_t$, le maximum $M_t$ et le minimum $m_t$ consiste à utiliser deux variables $E$ et $F$ ; $E$ (resp. $F$) vaut 1 si et seulement si $I_t$ est inférieure à $m_t$ (resp. $I_t$ est supérieure à $M_t$). Nous utilisons deux bascules $D$ et $G$ afin de savoir si le résultat des deux comparaisons est déjà connu ou non, le calcul se propageant des bits de poids forts aux bits de poids faibles. Dans le code rétine de la Table 3.7, $A$, $B$ et $C$ sont des variables temporaires. De même on note $X$, $MAX$ et $MIN$ les mots correspondants respectivement à $I_t$, $M_t$ et $m_t$.

Il ne nous reste plus qu'à mettre à jour les valeurs de $MIN$ et $MAX$ en fonction des valeurs respectives de $E$ et $F$. Nous notons que c'est le seul moment où $MIN$ peut décroître et $MAX$ peut croître puisque dans le reste nous ne faisons que soustraire des nombres négatifs au Minimum et ajouter des nombres négatifs au Maximum. Ici encore $B$ et $C$ sont des variables temporaires (voir Table 3.8).

Enfin, de la même manière que nous l'avons fait pour la moyenne récursive, il nous faut seuiller la différence $\Gamma_t = M_t - m_t$ en chaque pixel afin de décider quels sont ceux pour lesquels un mouvement est détecté.

### 3.5.4 Raffinements proposés

Pour les deux premiers algorithmes (moyenne récursive et gradient morphologique oublieux), nous utilisons un seuil global (i.e. le même pour tous les pixels). La valeur de ce seuil doit nécessairement être ajustée dynamiquement (localement et temporellement) si l'on souhaite que la détection soit adaptative.

$D = 0\,;\ E = 0\,;\ F = 0\,;\ G = 0\,;$
**Pour (i=7 ;i≥0 ;i- -) {**
$A = \bar{X}_i \wedge MIN_i\,;$
$B = X_i \wedge \overline{MIN_i}\,;$
$C = A \wedge \bar{D}\,;$
$E = E \vee C\,;$
$A = A \vee B\,;$
$D = A \vee D\,;$
$A = X_i \wedge \overline{MAX_i}\,;$
$B = \bar{X}_i \wedge MAX_i\,;$
$C = A \wedge \bar{G}\,;$
$F = F \vee C\,;$
$A = A \vee B\,;$
$G = A \vee B\,;$
**}**

TAB. 3.7 – Comparaison entre $X$, $MAX$ et $MIN$.

**Pour (i=0 ;i≤7 ;i++) {**
$B = E \wedge X_i\,;$
$C = \bar{E} \wedge MIN_i\,;$
$MIN_i = B \vee C\,;$
$B = F \wedge X_i\,;$
$C = \bar{F} \wedge MAX_i\,;$
$MAX_i = B \vee C\,;$
**}**

TAB. 3.8 – Calcul effectif de $MIN$ et $MAX$.

Nous utilisons le dialogue rétine-cortex pour adapter dynamiquement ce seuil en fonction d'une estimation de la distribution spatiale des différences. Pour réduire au maximum le flux d'information d'information, nous utilisons une somme sur une image binaire, en appliquant récursivement l'ajustement suivant : après avoir seuillé l'image à la valeur $\tau$, on calcule sur la rétine l'ensemble des points isolés (voir Chapitre 2 Section 2.3.4), puis on somme sur le cortex l'ensemble ainsi obtenu. Si la proportion de points isolés est inférieur à un taux fixé $\rho$ (typiquement de l'ordre de 1%), on décrémente $\tau$, sinon on l'incrémente.

| | | |
|---|---|---|
| $J_t$ | $=$ | $\delta_{\tau_t}(\Gamma_t)$ |
| $K_t$ | $=$ | ptisole$(J_t)$ |
| $S$ | $=$ | $\sum_{x \in l*L} K_t(x)$ |
| $\tau_{t+1}$ | $=$ | $\tau_t + 1$ si $S > \rho$ |
| $\tau_{t+1}$ | $=$ | $\tau_t - 1$ sinon |

TAB. 3.9 – Mise à jour de la valeur du seuil adaptatif $\tau$ en fonction de l'indice $\rho$ représentant typiquement 1% de la taille de l'image. $\Gamma_t$ représente le gradient morphologique oublieux utilisé en entrée de l'algorithme (il s'agit de la grandeur à seuillier).

La justification de cet ajustement repose sur l'hypothèse qu'un point isolé est généralement induit par du bruit et donc que le comptage de tels points fournit une estimation raisonnable de ce bruit. Ainsi l'on obtient une technique de seuillage global adaptatif qui offre d'assez bon résultats pour compenser un changement dans l'environnement comme par exemple, une modification des conditions de luminosité.

### 3.5.5 Résultats

La Figure 3.8 présente les résultats de la détection du mouvement utilisant les opérateurs morphologiques oublieux sur le banc algorithmique. La procédure de test employée pour générer les résultats est la même que celle présentée pour la moyenne récursive.

Bien que cela ne soit pas très visible sur les résultats proposés car l'amplitude du mouvement est déjà trop important pour les paramètres utilisés ($\alpha = 2$), les objets mobiles sont mieux segmentés et les phénomènes de fantômes très atténués. Cependant, les performances de cet algorithmes seront bien plus limitées lorsque l'on sortira de son domaine d'application , soit des mouvements lents, ou pour des objets mobiles de petites tailles, même en augmentant la valeur du terme oublieux $\alpha$ (on verra réapparaître alors le phénomène de fantôme) et ce même en utilisant une stratégie adaptative. De plus tous les types de déplacements ne seront pas aussi bien détectés, ce sera le cas en particulier pour des mouvements radiaux, plus difficile à segmenter.

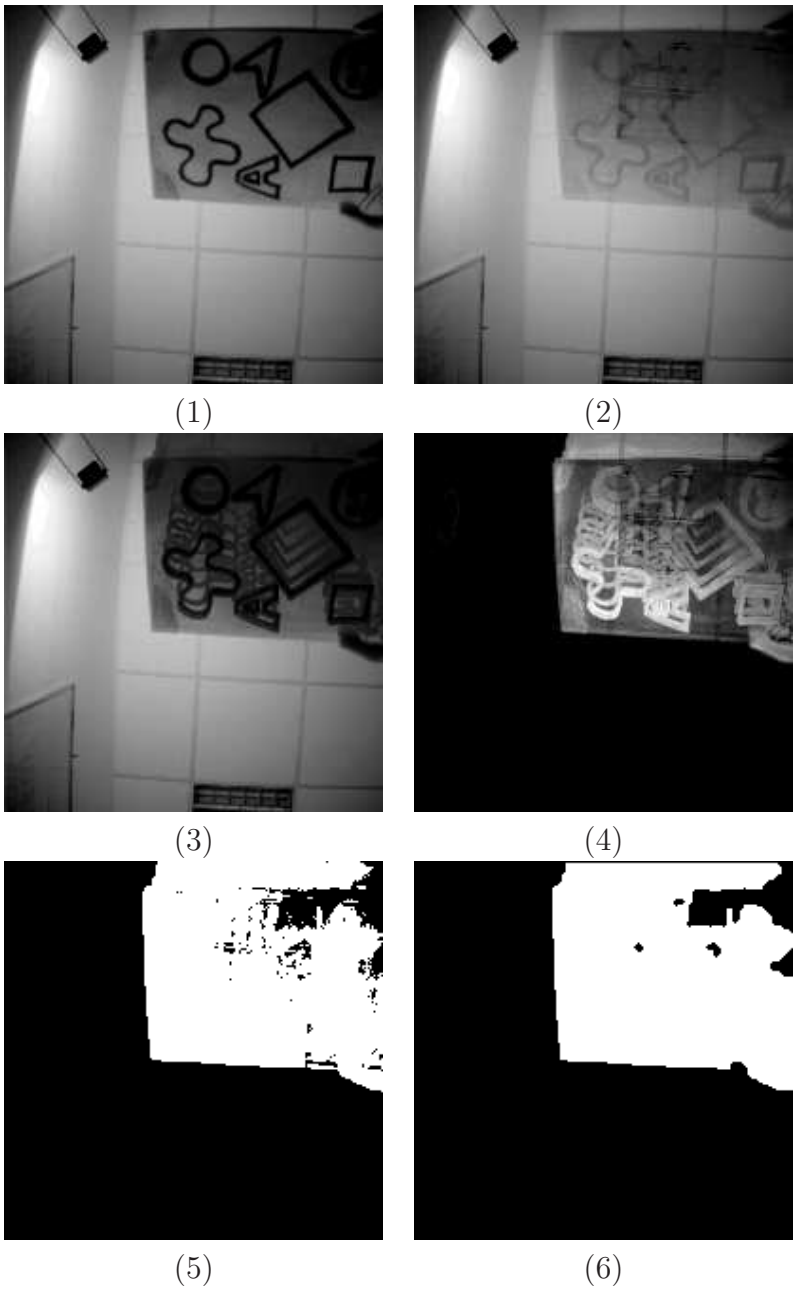


FIG. 3.8 – Calcul du gradient morphologique oublieux sur le banc algorithmique de la rétine :
(1) image acquise $I_t$ ; (2) Dilatation morphologique oublieuse $M_t$ ; (3) Erosion morphologique oublieuse $m_t$ et (4) gradient morphologique oublieux $G_t$ ; (5) Détection brute et (6) Détection filtree .

### 3.5.6 Conclusion

Les filtres morphologiques oublieux qui combinent opérations morphologiques et filtrages linéaires tirent parti des deux approches. En effet :

1. Les opérations morphologiques seules sont un puissant outil de détection dans la mesure où elles estiment l'amplitude des variations de niveau de gris dans un intervalle temporel donné ; par contre elles maximisent également le bruit temporel et les effets de "bougé"[14].
2. *A contrario*, le filtrage linéaire seul permet de limiter les effets du bruit temporel et de "bougé", mais il a un pouvoir de détection moindre.

Ainsi, la détection de mouvement à base de filtre morphologique oublieux est particulièrement adaptée aux mouvements de faible amplitude, c'est-à-dire à de petits objets, à des mouvements lents ou à des objets faiblement contrastés. Cependant, la segmentation des objets détectés n'est pas très précise sans l'adjonction d'un filtrage supplémentaire (voir Section 3.7).

## 3.6 L'estimation $\Sigma - \Delta$

### 3.6.1 Introduction

Cette méthode appartient à la catégorie des techniques par soustraction de fond statique. Elle ne nécessite qu'un seul paramètre pouvant être fixé de façon automatique (nous le montrerons par la suite). Son principe est de calculer une carte de l'activité temporelle locale, de manière à définir automatiquement les seuils de décision sur la classification fond/objet mobile, ces seuils étant variables et adaptés à la fois spatialement et temporellement. La détection est fondée sur le filtre $\Sigma$-$\Delta$, qui présente à la fois la robustesse liée à la non-linéarité, et une très bonne adéquation à la rétine numérique. Il s'agit de transposer une méthode électronique, donc bien adapté aux types de calcul rétiniens, au traitement d'images.

La dénomination du filtre vient du lien avec la modulation $\Sigma$-$\Delta$, utilisée en électronique pour la conversion analogique-numérique (CAN) : Nous comparons le signal analogique $a$ avec sa version numérisée $d$, si $d < a$, nous incrémentons $d$, sinon nous le décrémentons. Il a déjà été employé en détection de mouvement [MS95b] mais uniquement dans le cadre d'une approximation rapide du médian (en fait la valeur de la moyenne $\Sigma$-$\Delta$ se situe entre

[14] légers tremblements du capteur par exemple

la moyenne arithmétique et le médian, souvent un peu plus prche du médian). Il s'agit donc d'une technique itérative : $d(t+1) = d(t) + 1$ si $d(t) < a$ et $d(t+1) = d(t) - 1$ si $d(t) > a$. Lorsque le signal $a = a(t)$ varie avec $t$, la précision de la CAN n'est pas affectée si $|a'(t)| < 1$, où $a'$ désigne la dérivée de $a$ par rapport à $t$, sinon le biais du CAN $|a(t) - d(t)|$ est proportionnel à $|a'(t)|$. C'est principalement cette propriété que nous exploitons dans le filtre $\Sigma - \Delta$.

D'un point de vue théorique, on retrouve cette notion dans la moyenne récursive temporelle, l'incrément dépend linéairement de la différence entre l'évaluation et l'échantillon courant :

$$\begin{aligned} M_t &= \alpha * I_t + (1-\alpha) M_{t-1} \\ &= M_{t-1} + \alpha * (I_t - M_{t-1}) \\ &\simeq M_{t-1} + \delta_t(I_t) \end{aligned}$$

Si nous généralisons ceci, nous pouvons remplacer le taux d'apprentissage constant $\alpha$ par une fonction $\delta(x)$ pour changer la dépendance de l'incrément sur la différence. Par exemple, dans l'approximation par la plus précise gaussienne monomodale, $\delta(x)$ est choisie comme probabilité de l'échantillon courant, dans le modèle gaussien, la valeur de la fonction gaussienne centrée de variance $V_t$ (qui est estimé la même manière). Dans ce cas, la fonction d'incrment $C(x)$ tels que $M_t = M_{t-1} + C(I_{t-1})$ ressemble à la première dérivée de la fonction gaussienne $N(M_t, V_t)$. Il s'avre que cette approximation est en fait plus proche de l'évaluation de $\Sigma$-$\Delta$ (incrément constant) qu'à la moyenne récursive simple. Pour aller plus loin, si au lieu d'une distribution gaussienne, nous considérons une distribution de **Zipf-Mandelbrot**[15], alors dans ce cas $\delta(I_t - M_{t-1})$ devient un $K$ constant et la fonction d'incrément $C(x)$ ressemble maintenant à une sorte de fonction de **Heavyside**[16] décalée et centrée sur $M_{t-1}$, qui correspond à l'estimation de $\Sigma$-$\Delta$. D'un point de vue pratique, dans l'arithmétique à point fixe avec de petites dynamiques (ce qui est souvent le cas dans les systémes embarqués comme la rétine artificielle numérique), les méthodes d'approximation gaussiennes souffrent de truncatures et d'approximations numériques, dans lesquelles l'intervalle du taux d'apprentissage $(1/\alpha)$ est fortement limitée par la taille de l'arithmétique. Au contraire, l'évaluation de $\Sigma$-$\Delta$ n'implique aucune approximation numérique tout en préservant un degré maximal de précision, tant que

[15]une distribution où la probabilité diminue hyperboliquement par rapport à la distance à la moyenne

[16]La fonction de Heaviside, parfois appelée *unit step function* en anglais et nommée en l'honneur de Oliver Heaviside, est une fonction discontinue dont la valeur est zéro pour les arguments négatifs et un pour les arguments positifs :

$$H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{sinon} \end{cases}$$

La function est utilisée en mathématique en traitement du signal afin de repréesenter des signaux qui basculent à un moment spéecifié et restent dans set état indefiniment.

l'incrément constant $K$ est choisi comme $K = 1$ ( "1" représente en effet le "epsilon" ou la distinction minimale qui peut être faite dans le signal numérique).

La Figure 3.9 illustre cette approximation en présentant les trois fonctions mentionnées : une moyenne récursive temporelle avec $\alpha$ constant, une fonction gaussienne pour $\alpha = G_{M_t}^{\sqrt{(V_t)}}(I_t)$ et une fonction de heavyside pour $\alpha = Z_{M_t}(I_t)$.

### 3.6.2 Principe

Dans notre cas, le signal que nous voulons "estimer" n'est pas analogique, mais numérique, il s'agit de la séquence d'images incidentes $I_t$. Nous calculons pour chaque instant $t$, et pour chaque pixel $x$, le signal filtré $M_t$ (Table 3.10(1)). Si $I_t$ est un signal aléatoire quelconque, $M_t$ prend ses valeurs avec une probabilité maximale dans l'intervalle $[a, b]$ tel qu'il y a autant d'indices $\tau < t$ tels que $I_\tau < a$, que d'indices $\tau < t$ tels que $I_\tau > b$. Cette propriété confère au filtre $\Sigma$-$\Delta$ une bonne robustesse pour l'estimation du fond dans le cadre d'un point statique de la scène, pour lequel les variations hautes fréquences ont justement un caractère aléatoire (bruit d'acquisition).

Mais ce filtre a également une autre propriété qui nous intéresse pour la discrimination du mouvement. C'est le fait que la différence entre $I_t$ et $M_t$ soit proportionnelle au taux de variation de $I_t$. Or nous pouvons mesurer cette différence précisément (contrairement au cas où $I_t$ est analogique), et cette mesure nous fournit $\Delta_t$, qui est un indice de vraisemblance du mouvement (Table 3.10(2)).

Nous utilisons également le filtre $\Sigma$-$\Delta$ pour calculer la variance temporelle du signal $I_t$. Cette variance représente une mesure d'activité temporelle de chaque pixel, qui sera utilisée pour décider si le pixel est plutôt mobile ou fixe. Donc le second estimateur $V_t$ (Tableau 3.10(3)) utilisé dans la méthode a la dimension d'un écart-type temporel calculé comme le filtre $\Sigma$-$\Delta$ de la séquence des différences $\Delta_t$. Comme nous nous intéressons aux pixels dont le taux de variation est significativement supérieur à leur activité temporelle, nous appliquons le filtre $\Sigma$-$\Delta$ à $N$ fois les différences non nulles. Le paramètre $N$ est un paramètre de réglage s'apparentant à un gain. Son réglage n'est pas trop sensible et intervient plus à la mise en route afin d'accélerer la convergence de la moyenne $\Sigma$-$\Delta$.

Finalement, la fin de la détection au niveau pixel (traitements purement temporels) est réalisée en comparant $\Delta_t$ et $V_t$ (Tableau 3.10(4)). Le pixel $x$ est considéré mobile si $\Delta_t(x) > V_t(x)$, comme fixe sinon.

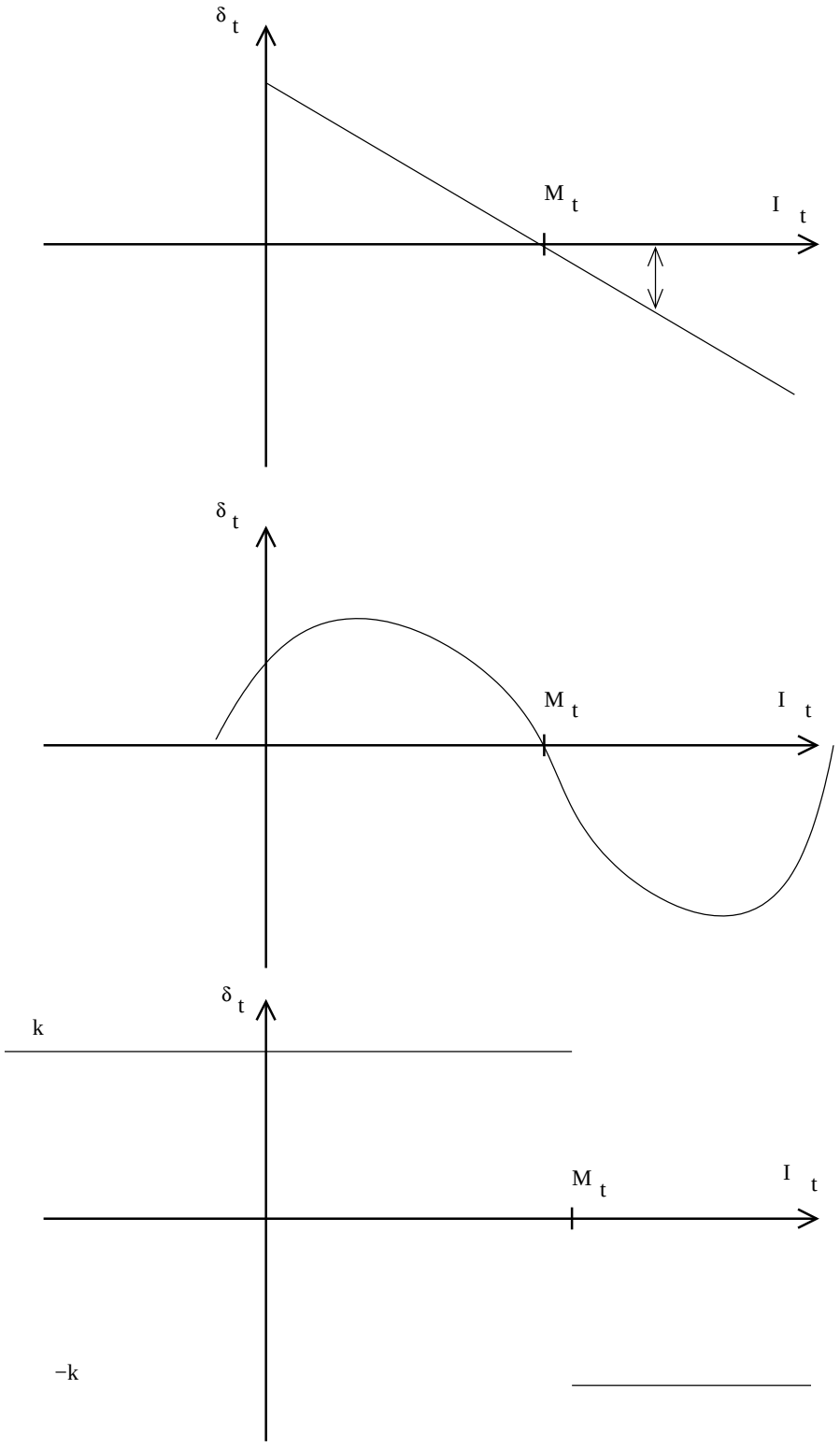



FIG. 3.9 – Une illustration des trois fonctions d'approximation gaussienne, en haut une moyenne récursive temporelle, au milieu une fontion gaussienne et en bas une fonction de Heavyside.

**Pour chaque trame t**
pour chaque pixel $x$ :
$\Delta_t(x) = M_t(x) - I_t(x)$

(1)

**Initialisation**
pour chaque pixel $x$ :
$M_0(x) = I_0(x)$
**Pour chaque trame t**
pour chaque pixel $x$ :
if $\Delta_t(x) < 0$, $M_t(x) = M_{t-1}(x) + 1$
if $\Delta_t(x) > 0$, $M_t(x) = M_{t-1}(x) - 1$

(2)

**Initialisation**
pour chaque pixel $x$ :
$V_0(x) = |\Delta_0(x)|$
**Pour chaque trame t**
pour chaque pixel $x$ tel que $\Delta_t(x) \neq 0$ :
if $V_{t-1}(x) < N \times |\Delta_t(x)|$, $V_t(x) = V_{t-1}(x) + 1$
if $V_{t-1}(x) > N \times |\Delta_t(x)|$, $V_t(x) = V_{t-1}(x) - 1$

(3)

**pour chaque trame t**
pour chaque pixel $x$ :
si $\Delta_t(x) < V_t(x)$
alors $D_t(x) = 0$
sinon $D_t(x) = 1$

(4)

TAB. 3.10 – L'estimation du fond par filtre Σ-Δ : (1) Calcul de la différence entre l'image originale et la moyenne. (2) Calcul de la moyenne Σ-Δ (mesure de vraisemblance du mouvement). (3) Calcul de la variance Σ-Δ définie par la moyenne Σ-Δ de $N$ fois les différences non nulles (paramétre de réglage). (4) Calcul des étiquettes de mouvement par comparaison entre la différence et la variance.

La Figure 3.10 illustre le comportement de l'algorithme pour trois pixels particuliers d'une séquence de 1000 images représentant une scène avec passage de véhicules, perturbé par un mouvement parasite au premier plan. Pour chaque pixel, nous avons représenté quatre graphes représentant les variations au cours du temps de $I_t$ (ligne continue rouge), $M_t$ (ligne avec des hachures espacées bleue), $\Delta_t$ (ligne hachurée verte) et $V_t$ (ligne pointillée violette). $D_t$ n'est pas explicitement représentée, mais correspond à la fonction indicatrice de $\{t; \Delta_t > V_t\}$.

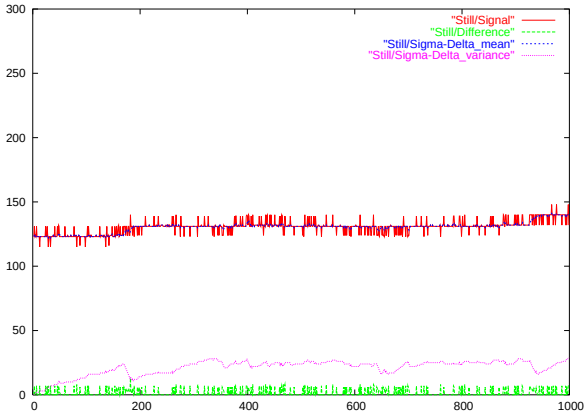
Le premier pixel (Figure 3.10(1)) correspond à une zone du fond statique sans perturbation. Les hautes fréquences correspondent aux bruits d'acquisition et de numérisation, qui sont intégrés et amplifiés au niveau de la mesure de variance, ce qui permet de définir un seuil minimal automatiquement[17]. Les basses fréquences correspondent soit à des mouvements lents d'objets faiblement contrastés[18], soit à des changements naturels d'illumination auxquels le fond s'adapte en permanence.

Le deuxième pixel (Figure 3.10(2)) correspond à une zone de passage d'objets mobiles[19]. Dans ce cas, le signal montre des pics plus ou moins importants au passage des objets. Ces changements importants ne sont pas pris en compte dans la moyenne $\Sigma$-$\Delta$, et donnent donc lieu à des différences importantes. Étant donné leur caractère transitoire, ces grandes valeurs ne sont que partiellement intégrées au niveau de la variance, le mouvement est donc détecté par discrimination des pics en comparant la différence et la variance.

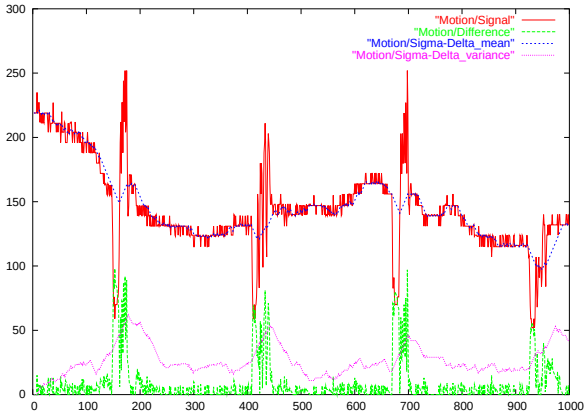
Le troisième pixel (Figure 3.10(3)) correspond à une zone de mouvement parasitaire, c'est-à-dire de changements physiques dues à la nature de la scène plutôt qu'à des objets mobiles[20]. Dans ce cas, le signal montre une répétition de pics d'assez grande amplitude. Mais le caractère répétitif de ces pics fait que la différence est prise entièrement en compte dans la variance $\Sigma$-$\Delta$, ce qui entraîne une augmentation locale du seuil de discrimination des objets mobiles.

La Figure 3.11 montre le résultat de cette méthode pour une trame prise dans une scène de trafic urbain. Les quatre images représentent respectivement $I_t$, $M_t$, $V_t$ et $D_t$. La discrimination des pixels "mobiles" correspond à la détection des pixels temporellement saillants au regard de l'activité temporelle. Cela permet d'écarter les mouvements non pertinents (bruit) mais aussi d'être moins sensible aux oscillations du capteur (voir Figure 3.17).

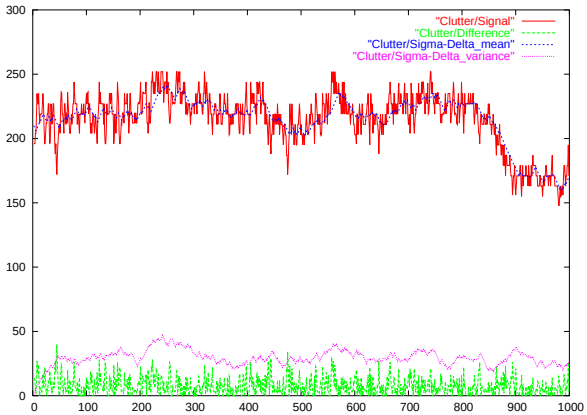
[17] ici $N = 4$
[18] une faible couverture nuageuse
[19] ici quatre véhicules en passage transversal
[20] ici des hautes herbes au premier plan qui se balancent sous l'effet de rafales de vent




(1)

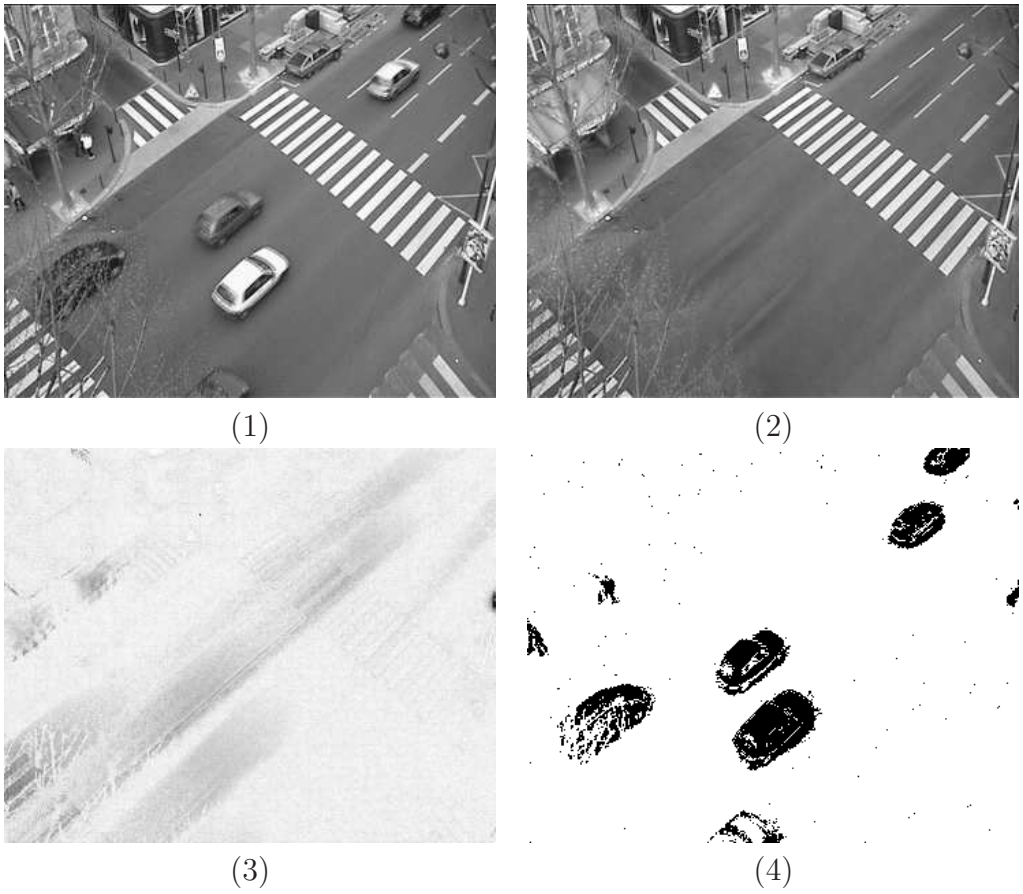



(2)




(3)

FIG. 3.10 – Comparaison du comportement de l'estimateur Σ-Δ en fonction du type de zone où se trouve le pixel : (1) Zone sans mouvement (2) Zone de mouvement (3) Zone de "clutter" (bruit de fond ou petits mouvements non pertinents). Un exemple de ce type de scène est proposé Figure 3 en Introduction Générale.



(1) (2)

(3) (4)

FIG. 3.11 – Résultat de l'algorithme pour une séquence de trafic urbain. (1) $I_t$ (2) $M_t$. (3) $V_t$ (avec une égalisation d'histogramme pour des raisons de visualisation). (4) $D_t$ (N=2). On peut voir que la segmentation des objets est précise quel que soit le type de mouvement observé, gros objets en mouvement rapide (voiture) ou petits objets lents (piétons en haut).

### 3.6.3 Implémentation rétinienne

La première étape de la moyenne Σ-Δ consiste à effectuer la différence signée entre l'image acquise $X_t$ et la moyenne $M_t$. En effet, plutôt que de comparer $X_t$ et $M_t$ nous effectuons leur différence et c'est le signe de différence qui nous permet de comparer ces deux mots.

Après avoir effectué la valeur absolue de cette différence nous effectuons véritablement, cette fois, la comparaison de $V_t$ et $N * |\Delta_t|$. Les variables $E$ et $F$ joue ce rôle dans l'algorithme de la Table 3.11 où $D$ est une bascule et $G$ une variable temporaire.

**Initialisation**
$D = 0$;
$E = 0$;
$F = 0$;
**Pour (i=7 ;i≥0 ;i- -) {**
$A = S_i \wedge \bar{V}_i$;
$B = V_i \wedge \bar{S}_i$;
$G = A \wedge \bar{D}$;
$E = E \vee G$;
$G = B \wedge \bar{D}$;
$F = F \vee G$;
$A = A \vee B$;
$D = D \vee A$;
**}**

TAB. 3.11 – Comparaison entre $V_t$ et $S_t = N * |\Delta_t|$.

Le rôle de $E$ est de nous indiquer si $V_t < N * |\Delta_t|$ et celui de $F$ est de savoir si $n * |\Delta_t| < V_t$. Il nous reste à mettre à jour la variance à l'aide des valeurs de $E$ et $F$. Ainsi nous décrémentons $V_t$ de $E$ et nous incrémentons $V_t$ de $F$ (voir Table 3.12).

A la différence des deux méthodes précédentes, nous utilisons ici uniquement un seuillage local, correspondant au résultat de la comparaison entre $\Delta_t$ et $V_t$.

Le seul paramètre visible de la méthode utilisant le filtre Σ-Δ est $N$, le nombre utilisé dans la comparaison de la variance $V_t$. L'échelle de valeur de $N$ est petite (entre 1 et 4), et généralement une puissance de 2 est choisie pour des raisons d'optimisation. De plus, ce paramètre peut-être automatiquement ajusté à l'aide d'une estimation du bruit. Cette estimation peut-être réalisée en comptant le nombre de points isolés dans l'image

Décrémentation de $E$
**Initialisation :**
$C = \bar{M}_0 \wedge E$
$M_0 = M_0 \oplus E$
**Pour (i=1 ;i≤7 ;i++) {**
$M_i = M_i \oplus C$
$C = C \wedge M_i$
**}**

Incrémentation de $F$
**Initialisation :**
$C = \bar{M}_0 \wedge F$
$M_0 = M_0 \oplus F$
**Pour (i=1 ;i≤7 ;i++) {**
$M_i = M_i \oplus C$
$C = C \wedge M_i$
**}**

TAB. 3.12 – Algorithme de la mise à jour de $M_t$.

du résultat de la détection $D_t$, sous l'hypothèse (classique) que de telles détections sont dûes au bruit. Cette technique de seuillage automatique est la même que celle employée pour les algorithmes précédents et présentée Section 3.4.4.

En fait, un autre paramètre moins évident est également présent. Il s'agit de la fréquence de rafraîchissement des statistiques $\Sigma$-$\Delta$. Cette fréquence a la dimension d'un nombre de niveaux de gris par seconde. il est clair qu'il doit être adapté à :

1. la **dynamique** de l'image (nombre de niveaux de gris) ;
2. la **fréquence** d'acquisition.

La robustesse de l'estimation $\Sigma$-$\Delta$ peut être encore améliorée en rafraîchissant $M_t$ seulement lorsque $\Delta_t < V_t$. Ce rebouclage est présenté en détail dans la partie suivante.

L'estimation du fond $\Sigma$-$\Delta$ est une méthode simple et efficace pour détecter les pixels changeant de manière significative dans une scène statique, par rapport à une constante de temps dépendant du nombre de niveaux de gris de l'image. Cependant il s'agit d'une méthode purement temporelle, qui produit une détection au niveau pixel. Nous présentons à la fin de ce chapitre (Section 3.7) quelques filtrages spatiaux afin d'améliorer et de régulariser les résultats de la détection.

**Rebouclage**

Un mécanisme de seuillage utilisant les points isolés (voir Chapitre 2 Section 2.3.4) peut également être utilisé dans le cas de l'algorithme (moyenne Σ-Δ) pour positionner automatiquement le paramètre $N$ du calcul $V_t$. Il convient de préciser que de telles mesures globales peuvent être calculées efficacement sur la rétine en utilisant des opérateurs intégraux analogiques tels que des mesures de courant consommé [BP97].

Une fois la détection purement temporelle $D_t$ calculée, on la régularise en exploitant les corrélations spatiales grâce au filtrage suivant : on élimine les petites composantes connexes de $D_t$ en utilisant l'ouverture par reconstruction :

$$L_t^{(0)} = Rec^{D_t}(\epsilon_{B_\lambda}(D_t))$$

où $\epsilon$ est l'érosion morphologique, $B_\lambda$ (l'élélement structurant), est une boule de rayon $\lambda$ et $Rec^Y(X)$ est la reconstruction géodésique de $X$ dans $Y$.

Finalement, on effectue une confirmation temporelle en calculant une autre reconstruction :

$$L_t = Rec^{L_t^{(0)}}(L_{t-1}^{(0)})$$

$L_t$ représente l'étiquette finale, le résultat de la détection. Sa sémantique est : les objets "plus gros" que $\lambda$ qui apparaissent sur deux trames consécutives.

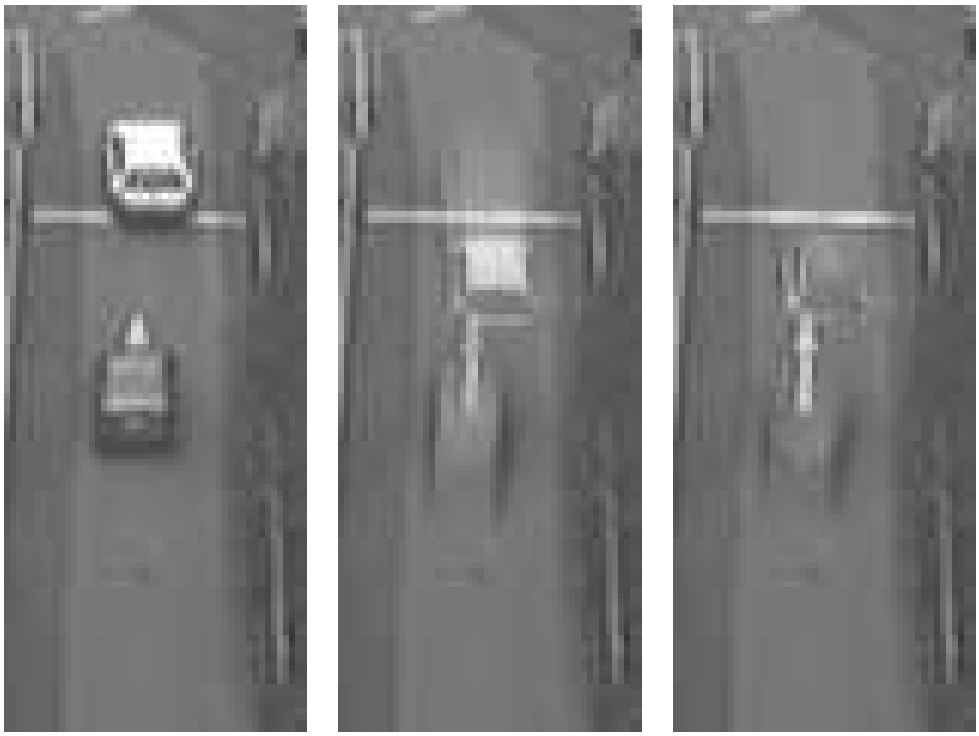
D'autre part, la détection est plus robuste si le fond n'est pas modifié pour les pixels correspondant à des objets en mouvement. Nous adoptons alors cette stratégie en rafraîchissant la moyenne Σ-Δ seulement lorsque $L_t(x) = 0$. Cela implique un réordonnancement de l'algorithme comme indiqué sur la Table 3.13.

Les effets du rebouclage sont illustrés sur la Figure 3.12. Nous pouvons y observer un détail d'une scène avec deux voitures roulant après s'être arrêtées à un feu rouge.

Il est plus sûr, à ce niveau de filtrage, d'appliquer le rebouclage seulement sur la moyenne $M_t$ au lieu des deux estimateurs $M_t$ et $V_t$ afin de ne pas avoir de faux objets détectés qui s'incrustent dans le fond. Le rebouclage peut-être utilisé à la fois sur $M_t$ et $V_t$ si une confiance de plus haut niveau est atteinte pour les objets détectés.

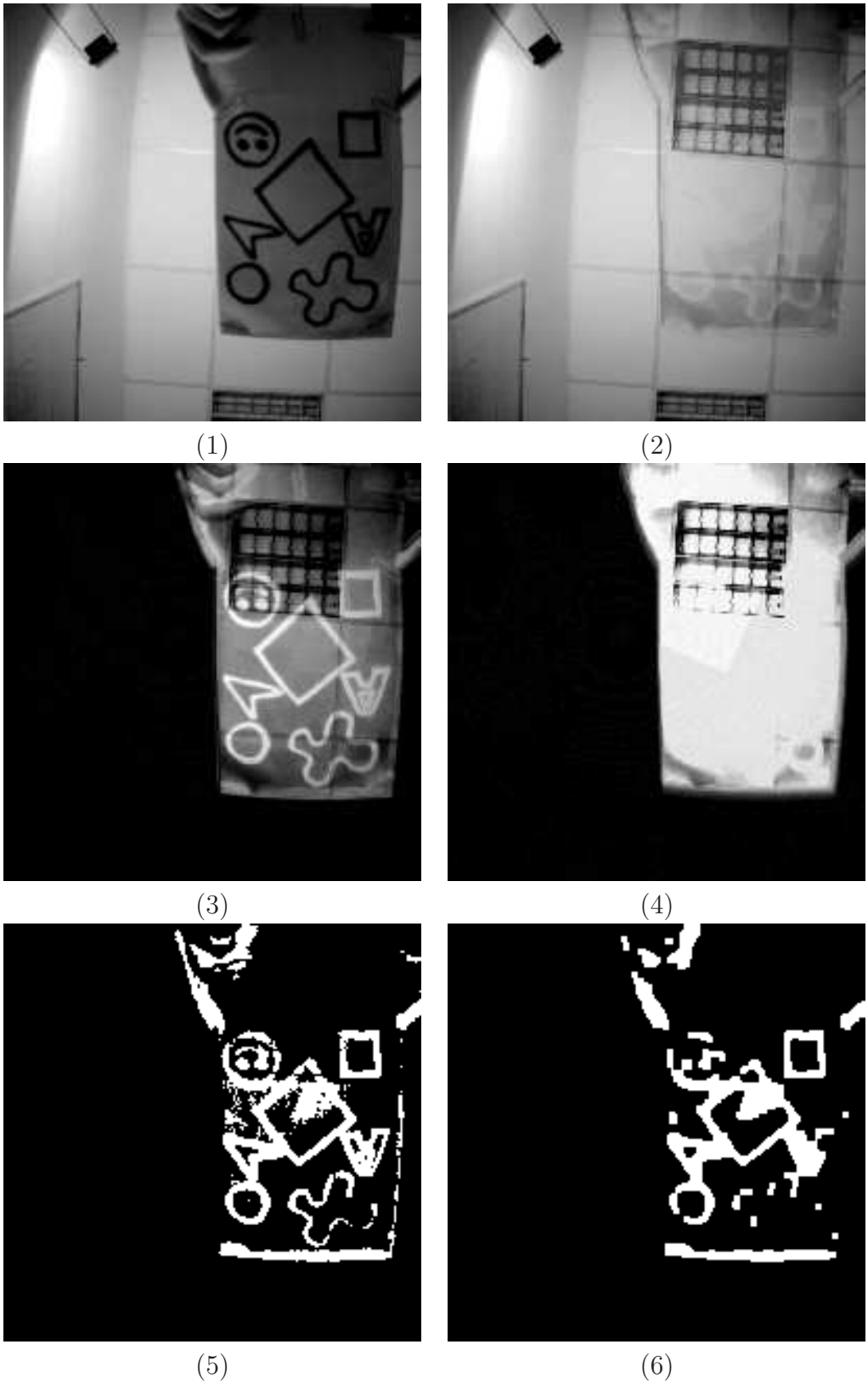
**Différence signée**
$S_t = M_t - I_t$
**Mise à jour de la variance**
si $|S_t| \neq 0$ :
si $V_{t-1} < N \times |S_t|$, $V_t = V_{t-1} + 1$
si $V_{t-1} > N \times |S_t|$, $V_t = V_{t-1} - 1$
**Reconstruction hybride des contours**
$\Delta'_t = HRec_{\alpha}^{|S_t|}(Min(||\nabla(I_t)||, ||\nabla(|S_t|)||))$
**Détection temporelle**
si $\Delta'_t < V_t$
alors $D_t = 0$
sinon $D_t = 1$
**Traitements binaires spatio-temporelles**
$L_t^{(0)} = Rec^{D_t}(\varepsilon_{B_\lambda}(D_t))$
$L_t = Rec^{L_t^{(0)}}(L_{t-1}^{(0)})$
**Mise à jour de la moyenne avec le rebouclage**
if $L_t = 0$ :
if $S_t < 0$, $M_t = M_{t-1} + 1$
if $S_t > 0$, $M_t = M_{t-1} - 1$

TAB. 3.13 – L'algorithme complet de la détection $\Sigma$-$\Delta$ avec rebouclage (la reconstruction hybride utilisée ici sera présentée par la suite Section 3.7.3 ).



(1) (2) (3)

FIG. 3.12 – L'effet de la boucle de rétro-action . (1) image originale (2) le fond $\Sigma$-$\Delta$ (3) fond avec rebouclage



(1) (2)

(3) (4)

(5) (6)

FIG. 3.13 – Résultats de la détection à l'aide de l'estimation Σ-Δ sur le banc algorithmique :
(1) image non traitée $I_t$ ; (2) la moyenne Σ-Δ $M_t$ ; (3) la variance Σ-Δ $V_t$ ; (4) la différence à la moyenne $\Delta_t$ ; (5) la détection brute $D_t$ ; (6) la détection filtrée $f(D_t)$.

### 3.6.4 Résultats

La Figure 3.13 présente les résultats de la détection du mouvement utilisant l'estimation Σ-Δ sur le banc algorithmique. La procédure de test employée pour générer les résultats est la même que celle présentée pour les algorithmes précédents.

### 3.6.5 Conclusion

Cet algorithme permet une détection robuste et précise d'objets mobiles pour un coût modéré en terme de consommation de mémoire et de complexité de calcul. Nous avons souligné les propriétés intéressantes que procure le filtre Σ-Δ pour la détection de paramètres saillants dans le signal temporel, montrant que l'intérêt de tels filtres va bien au-delà de ses propriétés de convergence au médian temporel.

Nous proposons une stratégie de régularisation spatio-temporelle, en utilisant une méthode hybride de reconstruction et de morphologie binaire spatio-temporelle afin d'exploiter la corrélation spatiale et accroître la confiance en la détection Σ-Δ. Nous pouvons aussi utiliser d'autres méthodes de régularisation comme une modélisation markovienne (voir section 3.7 pour plus de détails).

Les limitations de cette approche tient dans sa capacité d'adaptation à certaines scènes complexes comme cela est le cas pour des mouvements périodiques de forte amplitude (comme dans le cas de vagues en arrière-plan) qui seraient interprétés par l'algorithme comme faisant parties de l'ensemble des objets mobiles "intéressants" au lieu du fond si leurs périodes est trop longue. Nous avons étudié la possibilité de passer outre cette limite en combinant différents modèles de variance en utilisant différentes périodes d'échantillonnage pour l'estimation Σ-Δ afin d'obtenir une estimation plus riche quantitativement de l'activité du mouvement.

Le filtre Σ-Δ, entièrement non linéaire, permet d'avoir une plus grande robustesse dans les estimations du calcul du fond et semble le plus adapté au jeu d'instruction minimaliste de la rétine. Il permet en outre de calculer plusieurs ordres de statistiques temporelles à dynamique constante, ce qui est beaucoup plus délicat à envisager avec les techniques linéaires comme nous le verrons dans la Section 3.7.5.

## 3.7 Améliorations apportées aux algorithmes sur architecture classique

### 3.7.1 Introduction

Dans cette section, nous apportons des améliorations aux algorithmes présentés dans le cadre d'une implémentation sur une architecture classique. Ces raffinements n'ont pas fait l'objet d'une implémentation sur le circuit rétinien soit par manque de temps (et parce que d'autres développements nous paraissaient plus importants, soit parce qu'ils ne sont tout simplement pas possible actuellement sur la rétine.

Ces améliorations sont de quatre formes :

1. un mécanisme de paramétrage automatique du terme linéaire des opérateurs morphologiques temporels oublieux ;
2. des filtrages spatio-temporels permettant une meilleure segmentation des résultats de l'estimation Σ-Δ ;
3. l'utilisation des opérateurs Σ-Δ avec la modélisation markovienne pour la détection du mouvement ;
4. l'utilisation des opérateurs Σ-Δ dans une estimation multimodale du fond.

### 3.7.2 Paramétrage automatique du terme oublieux $\alpha$

Nous présentons une stratégie mise en place en simulation afin de fixer automatiquement le terme linéaire utilisé dans les opérateurs morphologiques temporels oublieux à l'aide d'un estimateur basé sur un filtre gaussien. Cette technique vient s'ajouter à la détection du mouvement présentée dans la section 3.5.

Nous utilisons le filtre Σ-Δ présenté dans la section précédente afin de calculer la variance $V_t$ de la séquence. Nous calculons ensuite le filtre Σ-Δ de $N$ fois les valeurs nonnulles du filtre morphologique oublieux. plus précisément, l'estimation locale de l'activité spatio-temporelle est définie par :

$$\Theta_t = G_\sigma(V_t)$$

où $G_\sigma$ est le filtre gaussien bidimensionnel d'écart-type $\sigma$ définie par :

$$G_\sigma(x,y) = \frac{1}{2\Pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Le lissage par un filtre gaussien est justifié puisque nous ne connaissons rien *a priori* sur les interactions entre pixels de l'image. Une justification mathématique de l'emploi d'un lissage par une gaussienne a été formulée par Canny [Can86], qui montre que la dérivée première d'un filtre gaussien fournit une valeur proche de la valeur optimale pour le critère par lequel il se propose de définir un bon détecteur de contours.

$\Theta_t$ est utilisée comme première étape de décision pour le label "en mouvement" de chaque pixel (on rappelle que $\Gamma_t$ représente le gradient morphologique oublieux présenté Section 3.5) :

$$D_t = 1 \text{ si } \Gamma_t > \Theta_t \text{ et } D_t = 0 \text{ sinon.}$$

$\Theta_t$ est aussi utilisée pour mettre à jour le terme oublieux en chaque pixel. Inspiré par [Pic04] qui utilise un taux d'apprentissage adapté localement pour le calcul de l'estimation du fond récursif, nous calculons un $\alpha$ local pour le pixel $x$ par la formule suivante :

$$\alpha_t(x) = 2^{-\frac{\overline{\Theta_t(x)}^2}{k^2}}$$

où $\overline{\Theta_t}$ est le complémentaire de $\Theta_t$ (e.g $\overline{\Theta_t} = 255 - \Theta_t$ pour des images codées en 256 niveaux de gris) et $k$ est une constante utilisée pour dimensionner la valeur de $\alpha$.

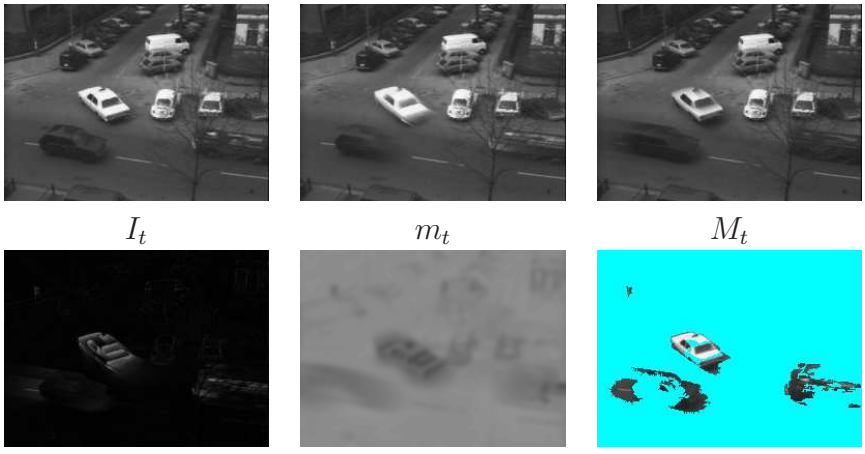



FIG. 3.14 – Résultats de la détection à l'aide du filtre morphologique oublieux pour la séquence du taxi de Hambourg (trame $n = 20$). Les paramètres utilisés sont $N = 2$ (variance) pour $V_t$, $\sigma = 2.5$ pour l'écart-type du filtre gaussien utilisé pour calculer $\Theta_t$, $k = 140$ comme constante utilisé dans le calcul de $\alpha_t$.

Pour utiliser ces filtres dans un cadre de détection du mouvement, nous avons besoin d'une règle de décision afin de discriminer les objets en mouvement du fond. La scène étant constamment en évolution - typiquement sous des changements d'illumination ou

de conditions climatiques - le critère de décision doit être adaptatif temporellement. En outre la variation temporelle, due probablement aux objets en mouvement, mais aussi au bruit ou au mouvement non pertinent, n'est pas uniformément distribuée dans toute la scène. Aussi la décision doit être différenciée localement.

Dans notre algorithme, nous calculons, à l'aide du filtre morphologique oublieux temporel, une estimation locale de l'activité spatio-temporelle. Cette estimation est utilisée à la fois pour décider au niveau pixel de l'étiquette de mouvement et pour ajuster la valeur du terme oublieux $\alpha$.

Le terme d'oubli $\alpha$ est ainsi localement ajusté de telle façon que les filtres oublieux utilisent un terme de mémorisation long dans des régions de faible activité spatio-temporelle et un terme de mémorisation court dans les régions de haute activité spatio-temporelle. Ainsi, la détection d'objets en mouvement lent, petits objets ou d'objets faiblement contrastés est accentuée, alors que les objets en mouvement ample et fortement contrastés sont mieux segmentés.

La figure 3.14 montre les différentes étapes de l'algorithme de détection du mouvement appliqué à la séquence du Taxi de Hambourg. La dernière image de la figure représente $D_t$, le résultat de la détection après avoir retiré les plus petites régions en utilisant une ouverture par reconstruction de boule de rayon unitaire [Vin93].

Nous remarquons que les mouvements de faible amplitude comme le piéton en haut à gauche de la scène ainsi que les objets en mouvement faiblement contrastés comme la voiture sombre en bas à gauche de l'image sont bien détectés alors que le taxi très contrasté au centre est correctement segmenté. C'est l'effet de l'adaptabilité du terme de mémorisation $\alpha$ des filtres oublieux, qui ajuste la capacité de détection des filtres oublieux à la quantité de mouvement.

### 3.7.3 Filtrages spatio-temporels de l'estimation Σ-Δ

Le filtrage spatio-temporel que nous présentons ici a un triple objectif :

1. éliminer les pixels non significatifs de la détection (bruit, fausse détection), et améliorer la segmentation des objets en mouvement ;
2. réduire les effets dus au phénomène de "fantôme" qui produisent de fausses détections aux endroits où les objets mobiles se remettent en mouvement après être restés un long moment fixes ;
3. réduire les problèmes d'ouverture qui provoquent des détections de piètre qualité

pour les objets dont le mouvement projeté est faible[21].

La première partie des filtrages spatiaux est composée d'opérations en niveaux de gris. Les entrées du filtrage sont l'image originale $I_t$ et la différence $\Sigma$-$\Delta$ d'image $\Delta_t$. L'objectif de ce module est d'éliminer les objets fantômes dans $\Delta_t$ en les discriminant dans $I_t$. Cette opération est menée grâce à la formule suivante :

$$\Delta'_t = HRec_{\alpha}^{\Delta_t}(Min(||\nabla(I_t)||, ||\nabla(\Delta_t)||))$$

Cela correspond à la reconstruction dans $\Delta_t$ de l'image des minima entre le module du gradient de $\Delta_t$ et le module du gradient de $I_t$. La sémantique de cette formule est : "$\Delta'_t$ est construite avec les composantes de $\Delta_t$ qui sont aussi dans $I_t$".

Les modules des gradients de $\Delta_t$ et de $I_t$ sont calculés en estimant les composantes des dérivées premières par convolution d'un masque de Sobel, et en calculant alors la norme Euclidienne du vecteur. Nous effectuons ensuite le calcul de l'image minimum $Min$, définie en chaque pixel $x$ par $Min(I_1, I_2)(x) = min(I_1(x), I_2(x))$. Cette opération agit comme une conjonction logique, retenant seulement les contours d'un objet qui sont présent à la fois dans $\Delta_t$ et dans $I_t$.

Afin de reconstruire l'objet entier dans $\Delta_t$, et pas seulement ses contours, nous effectuons une reconstruction des contours communs $Min(||\nabla(I_t)||, ||\nabla(\Delta_t)||)$ dans $\Delta_t$.

La première idée pourrait-être d'utiliser la reconstruction géodésique (Chapitre 2, Définition 51), définie par la relaxation de la dilatation géodésique $\delta_B^Y(X) = Min(\delta_B(X), Y)$ ($\delta$ est ici la dilatation morphologique, $B$ l'élément structurant définissant la topologie, $Y$ l'image de référence et $X$ l'image marqueur). En fait la reconstruction géodésique n'est pas adaptée ici car le critère de connexion seul n'est pas assez robuste et dans la plupart des cas, les objets et leurs fantômes sont tous deux reconstruits.

### Reconstruction hybride

Nous préférons utiliser des opérations de reconstructions hybrides, basés sur les opérateurs morphologiques oublieux (voir Chapitre 3.3.5). Nous allons d'abord définir la dilatation hybride comme une version spatiale de la dilatation oublieuse, calculée par la séquence causale :

$$HDil_{\alpha}(I)^{(0)}(x, y) = \alpha I(x, y) + (1 - \alpha) \max(I(x, y), HDil_{\alpha}(I)^{(0)}(x - 1, y))$$

[21] comme les objets en mouvement radiaux

suivi par la séquence anticausale :

$$HDil_\alpha(I)^{(1)}(x,y) = \alpha HDil_\alpha(I)^{(0)}(x,y) + (1-\alpha)\max(HDil_\alpha(I)^{(0)}(x,y), HDil_\alpha(I)^{(1)}(x+1,y))$$

puis la séquence causale verticale :

$$HDil_\alpha(I)^{(2)}(x,y) = \alpha HDil_\alpha(I)^{(1)}(x,y) + (1-\alpha)\max(HDil_\alpha(I)^{(1)}(x,y), HDil_\alpha(I)^{(2)}(x,y-1))$$

et enfin :

$$HDil_\alpha(I)(x,y) = \alpha HDil_\alpha(I)^{(2)}(x,y) + (1-\alpha)\max(HDil_\alpha(I)^{(2)}(x,y), HDil_\alpha(I)(x,y+1))$$

ici $\frac{1}{\alpha}$ a la dimension d'un rayon spatial et la paramètre $\alpha$ tient donc lieu d'élément structurant.

La reconstruction hybride est donc basée sur le même schéma en utilisant la séquence :

$$HRec_\alpha^J(I)^{(0)}(x,y) = \min(J(x,y), \alpha I(x,y) + (1-\alpha)\max(I(x,y), HRec_\alpha^J(I)^{(0)}(x-1,y))$$

et ainsi de suite, jusqu'à convergence ou pour un nombre d'itérations fixé préalablement.

Le comportement de la reconstruction hybride peut-être observé sur la Figure 3.15, où les objets (voitures, piétons) bougent après s'être complètement immobilisés pendant environ 20 trames consécutives. La reconstruction hybride dans cette application possède l'avantage d'"oublier" graduellement (exponentiellement en fait) le marqueur qui joue le rôle de fonction de confiance, au lieu d'être strictement basée sur la connexité.

Nous pouvons observer que la qualité de l'étape de marquage des contours dépend du contraste du fond lui-même, (voir Figure 3.15(7)-(10)). Nous nous sommes focalisés ici sur des cas extrêmes, les objets sont complètement fixes et incrustés dans le fond au début de la séquence. De plus, le rebouclage que nous allons examiner dans la suite peut aussi réduire de façon significative les effets du phénomène de fantôme.

### Morphologie spatio-temporelle binaire

Après avoir calculé la reconstruction hybride, nous réalisons un seuillage adaptatif sur $\Delta'_t$ (au lieu de $\Delta_t$ dans la version purement temporelle) :

$$\text{Si } \Delta'_t > V_t \text{ alors } D_t = 1 \text{ sinon } D_t = 0$$

Nous éliminons alors les petites composantes connexes de $D_t$ en utilisant l'ouverture par reconstruction :

$$L_t^{(0)} = Rec^{D_t}(\varepsilon_{B_\lambda}(D_t))$$

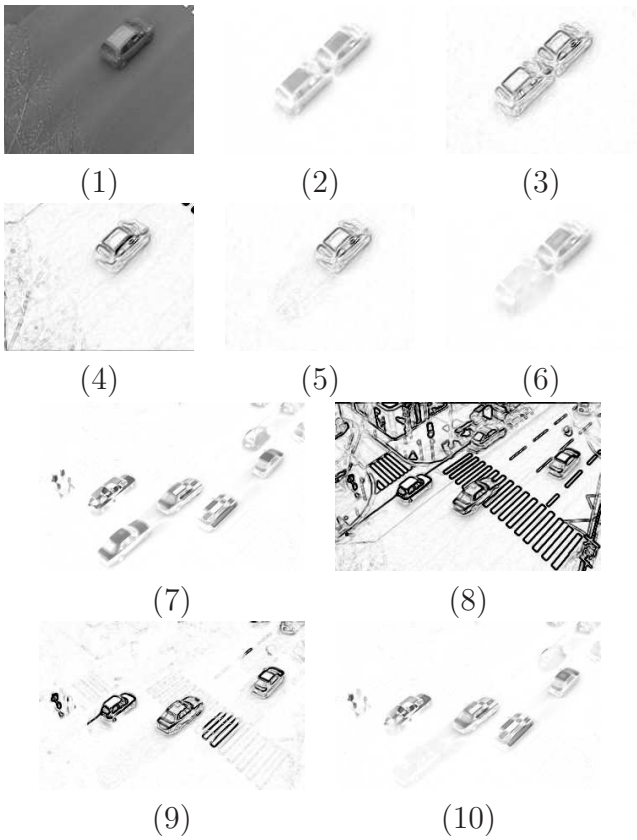



FIG. 3.15 – Phénomènes de fantôme éliminés par la reconstruction hybride des contours communs. Première exemple (1 voiture) : (1) $I_t$ (image originale) (2) $\Delta_t$ (différence à la moyenne $\Sigma$-$\Delta$) (3) $||\nabla(\Delta_t)||$ (module du gradient de la différence) (4) $||\nabla(I_t)||$ (module du gradient de $I_t$) (5) $Min(||\nabla(\Delta_t)||, ||\nabla(I_t)||)$ (minimum des modules des gradients) (6) $\Delta'_t$ (reconstruction hybride dans $\Delta_t$ de l'image des minima). Second exemple (5 voitures, 1 piéton) : (7) $\Delta_t$ (8) $||\nabla(I_t)||$ (9) $Min(||\nabla(\Delta_t)||, ||\nabla(I_t)||)$ (10) $\Delta'_t$.

Où $\varepsilon$ est l'érosion morphologique, $B_\lambda$, l'élément structurant, est une boule de rayon $\lambda$.

Finalement, nous réalisons une confirmation temporelle en calculant une autre reconstruction :

$$L_t = Rec^{L_t^{(0)}}(L_{t-1}^{(0)})$$

$L_t$ représente le label final, le résultat de la détection. Sa sémantique est : les objets "plus gros que" $\lambda$ qui apparaissent sur 2 trames consécutives. Ceci est illustré sur la Figure 3.16, en utilisant $\lambda = 1$, dans une scène représentant deux personnes en mouvement de marche.

(1) (2) (3)

FIG. 3.16 – Morphologie spatio-temporelle binaire. (1) $D_t$ (2) $L_t^{(0)}$ (3) $L_t$

### 3.7.4 Régularisation Markovienne

Nous suivons le modèle qui a été utilisé pour l'exécution en temps réel sur différentes architectures [CDLC96] et [LLMG99]. Ce modèle de Markov est basé sur l'évaluation d'un champ de déplacement binaire (avant-plan / arrière-plan) $e$ donnant un **champ d'observation** $o$, en maximisant un critère de **maximum *a posteriori*** bayésien : étant donné une réalisation du champ d'observation $o = y$, trouver la réalisation $x$ du champ d'étiquette de mouvement $e$ qui maximise la probabilité conditionnelle $P(e = x|o = y)$. Sous l'hypothèse que $e$ est un champ de Markov, et un modèle probabiliste liant $o$ et $e$, cela correspond à trouver le champ de mouvement $e$ qui minimise la fonction globale d'**énergie** définie pour l'ensemble de pixels $\mathbb{S}$ comme suit :

$$\begin{aligned} U &= \sum_{s\in\mathbb{S}}[U_m(e(s)) + U_a(e(s), o(s))], \\ \text{avec } U_m(e(s)) &= \sum_{r\in\mathcal{V}(s)} V_e(e(s), e(r)), \\ \text{et } U_a(e(s), o(s)) &= \tfrac{1}{2\sigma^2}[o(s) - \Psi(e(s))]^2. \end{aligned}$$

$U_m(e(s))$ est appelée l'**énergie du modèle** et fournie la régularité spatio-temporelle dans le domaine du mouvement. Elle est basée sur la modélisation Markovienne de $e$

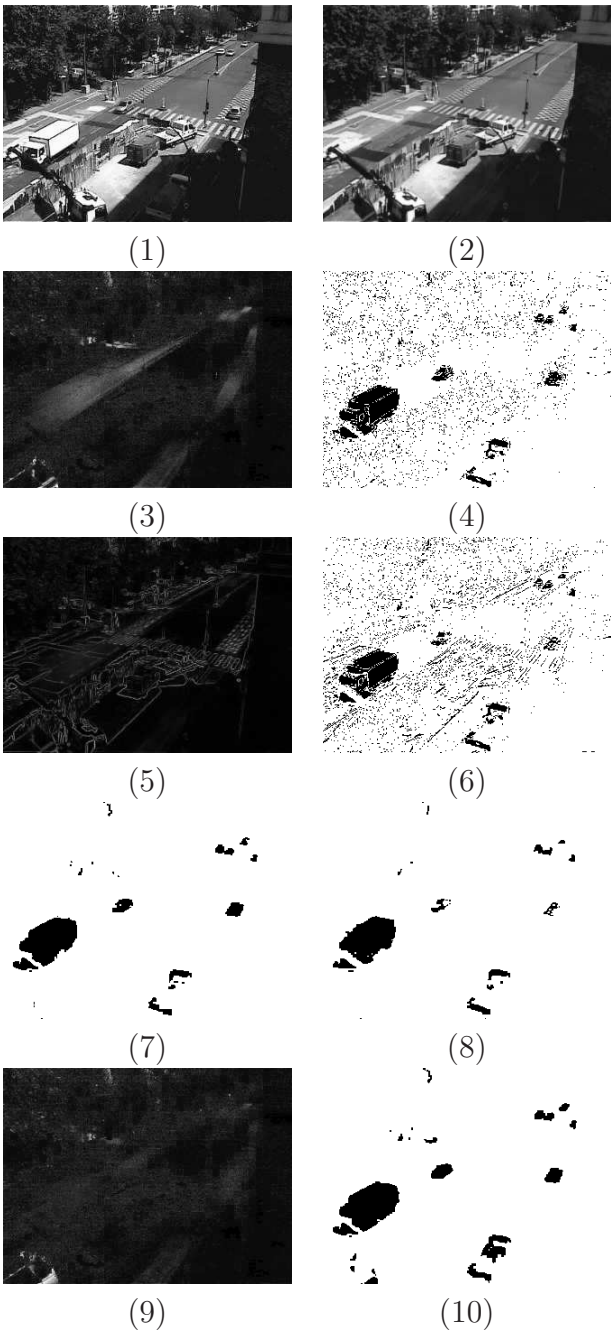



FIG. 3.17 – Résultat de l'estimation Σ-Δ avec régularisation Markovienne sur une séquence de trafic urbain. (1) $I_t$ (2) $M_t$. (3) $V_t$ (affiché avec un histogramme normalisé). (4) $D_t$ (N=2). (5) $V_t$ avec des oscillations simulées de la caméra (histogramme normalisé). (6) $D_t$ pour une caméra oscillante (N=2). (7) Détection après régularisation Markovienne (5 itérations). (8) *idem* pour une caméra oscillante. (9) $V_t$ en utilisant le rebouclage (en appliquant la même transformation que pour l'image (3). (10) Détection en utilisant le rebouclage sur le calcul niveau pixel et régularisation Markovienne.

comme champ de Gibbs, où $\mathcal{V}$ est l'ensemble de voisins du pixel $s$, et les fonctions de potentiel $V_e(e(s), e(r))$ sont égales à $-\beta_{sr}$ si $e(s) = e(r)$, et $+\beta_{sr}$ si $e(s) \neq e(r)$. Les $\beta_{sr}$ sont des constantes positives, dont les valeurs dépendent de la nature du voisinage. Nous utilisons une topologie spatio-temporelle uniforme 6-connectée avec 3 valeurs différentes $\beta_s = 20$ pour les 4 voisins spatiaux, $\beta_p = 10$ pour le voisin passé, et $\beta_f = 30$ pour le voisin futur.

$U_a(e(s), o(s))$ est l'**énergie d'adéquation**[22] et assure un certain niveau d'attachement aux données d'entrées, aux observations $o$. Ce terme vient de la probabilité conditionnelle du champ d'observation $o$, le long du champ de mouvement $e$, en supposant que $o(s) = \Psi(e(s)) + n(0, \sigma^2)$, avec $n(0, \sigma^2)$ un bruit gaussien centré de variance $\sigma^2$, $\psi(e(s)) = 0$ si $e(s)$ a la valeur de fond, et $\psi(e(s)) = \alpha$ si $e(s)$ a la valeur de premier plan. les auteurs utilisent la différence absolue entre deux trames consécutives comme champ d'observation. Ils emploient une valeur constante pour $\alpha$ (20), et estiment $\sigma^2$ en calculant la variance spatiale des observations.

La minimisation de l'énergie globale $U$ est obtenue par la relaxation déterministe appelée **ICM** : tous les pixels sont séquentiellement mis à jour, et chaque pixel $s$ donne l'étiquette $e(s)$ correspondant à la plus petite énergie locale $U_m(e(s)) + U_a(e(s), o(s))$. Habituellement, au lieu d'une relaxation vraie, un nombre limité de balayages est exécuté (en général 4). Cet algorithme est connu pour être très sensible à la qualité de la valeur initiale du champ estimé de mouvement. Les auteurs (voir [DMLM05] et [Dum96]) utilisent un seuillage de l'observation (c.-à-d. la différence absolue entre deux trames consécutives) comme évaluation initiale de $e$.

Dans notre algorithme, nous employons le même modèle, avec les exceptions suivantes :
- pour l'observation $o$, nous employons $\Delta$, la différence entre le fond et la trame courante ;
- nous employons la variance $\Sigma$-$\Delta$ $V$ comme deuxième champ d'observation, pour estimer localement le facteur de dispersion : $(\frac{v}{n})^2$ est utilisé au lieu de $\sigma^2$ pour pondérer l'importance relative de $U_a$ par rapport à $U_m$ ;
- l'initialisation de la relaxation Markovienne correspond à la détection niveau pixel $D$.

Quels sont les avantages de cet algorithme comparé au modèle Markovien original ?

D'abord, la différence avec le fond $\Sigma$-$\Delta$ est plus robuste que la différence trame à trame, parce qu'elle combine l'information sur une grande période au lieu de deux trames. Elle est beaucoup moins sensible au problème d'ouverture, qui rend difficile la détection de grandes zones homogènes en mouvement. Elle dépend également moins de la vitesse des

[22] *fitness* en anglais

objets.

Ensuite, pour les mêmes raisons, $D$ est en général un bien meilleur candidat pour initialiser la relaxation qu'une différence de trame binarisée, parce qu'il est plus proche de la solution prévue. Il est à noter que pour l'algorithme ICM, une fois l'initialisation calculée, les autres paramètres du modèle ne sont pas critiques, et ont montré de bons comportements sur un large panel de séquences.

Finalement, le paramètre de dispersion est calculé localement, aucun calcul global est nécessaire à chaque trame. Ceci permet le calcul de l'algorithme entier en utilisant seulement la mémoire locale, permettant de ce fait un parallélisme spatial massif.

La Figure 3.17 présente les résultats de l'estimation $\Sigma$-$\Delta$ avec régularisation Markovienne sur une séquence de trafic urbain.

La sortie de la régularisation Markovienne est une estimation bas niveau du premier plan, comprenant les pixels saillants temporels. Afin d'améliorer la qualité de la détection et d'abaisser le taux de fausses alarmes, certains traitements de niveau plus élevés sont nécessaires, en utilisant des calculs régionaux et globaux.

Les pixels sont regroupés en régions représentant les objets. Cela est habituellement réalisé par la fusion de composantes connexes. Les objets résultants subissent alors un filtrage morphologique, qui peut rejeter certains objets sous des critères de taille ou de forme. Un filtrage cinématique peut également être utilisé afin de distinguer les objets dont le mouvement est présent sur plusieurs trames consécutives (par exemple voiture, piéton, ...).

En plus de la diminution du taux de fausses alarmes, l'intérêt du traitement au niveau global est de permettre une rétroaction sur la détection de bas niveau. Un des exemples les plus immédiats est l'adaptation à un changement soudain de fond : si un index global de confiance du fond (par exemple la superficie relative occupée par le premier plan) diminue en deça d'un certain seuil, la décision peut être prise de réinitialiser le fond, afin d'abaisser le temps de réadaptation. Un autre exemple de rebouclage a été présenté et peut être ici aussi utilisé.

Une étude approfondie de la comparaison des performances de l'algorithme $\Sigma$-$\Delta$ complet et de son utilisation avec une modélisation markovienne sur plusieurs architectures

(sur processeur **PowerPC**[23] et sur la **maille associative**[24]) a été menée et montre l'intérêt de l'algorithme dans un cadre plus général [DMLM05].

### 3.7.5 Estimation Σ-Δ multimodale du fond

Les traitements spatio-temporels et les rebouclages, présentés dans les sections précédentes, permettent une amélioration sensible de la robustesse, dans le cas des objets ralentissant, à l'arrêt ou en déplacement radiaux. Néanmoins, l'estimateur Σ-Δ est caractérisé par une constante de temps : sa période de mise à jour, qui a la dimension du nombre de niveaux gris par seconde. Ceci induit une limitation de l'approche basique des possibilités d'adaptation dans certaines scènes complexes, typiquement dans le cas des scènes où de multiples objets de tailles et de vélocité différentes se croisent. Nous proposons dans cette section un cadre de généralisation de l'estimateur Σ-Δ au calcul de fonds multimodaux.

Le principe est de calculer, au lieu d'un fond simple $M_t$, un ensemble de $K$ fonds : $\{m_t^i\}_{1\leq i\leq K}$. Chaque fond $m_t^i$ est caractérisé par sa période de mise à jour $\alpha_i$ et par sa phase $\phi_i$. Un ensemble de $K$ variance $v_t^i$ est également calculé comme la moyenne Σ-Δ des différences entre $I_t$ et $m_t^i$. La décision fond / avant-plan est alors faite en comparant cet échantillon à chaque fond, auquel on attache une valeur de confiance qui est (1) proportionnelle à $\alpha_i$ et (2) inversement proportionnelle à $v_t^i$.

La Table 3.14 montre un exemple du calcul de fond multi-fréquences utilisant $K$ différentes périodes $\alpha_1 < \ldots < \alpha_K$. Les phases sont ignorées dans cet exemple. Les moyennes Σ-Δ $m_t^i$ peuvent être calculées récursivement : $m_t^i = m_{t-1}^i + sgn(m_t^{i-1} - m_{t-1}^i)$, avec la convention $m_t^0 = I_t$[25]. Dans cet exemple, nous calculons explicitement un fond de

[23]Le PowerPC est une architecture de microprocesseurs développée conjointement par Apple, IBM et Motorola. Elle utilise un modèle **RISC** (microprocesseur à jeu d'instruction réduit ou *reduced instruction-set computer*), privilégiant pipeline et jeu d'instructions réduit. L'architecture du PowerPC est directement dérivée de l'architecture POWER (Performance Optimization with Enhanced RISC) d'IBM.

Un haut degré de parallélisme permet d'effectuer parfois jusqu'à quatre ou cinq opérations simultanément, par exemple multiplication et cumul, test, et branchement conditionnel.

Egalement connue sous le nom de PPC, cette architecture est la plus utilisée de nos jours, notamment en informatique embarquée, en raison de sa faible consommation et de son haut parallélisme qui permet à puissance de traitement égale de travailler avec des fréquences d'horloge bien plus basses

[24]La maille associative (*associative mesh*) est une structure SIMD de processeurs construits sur l'observation des mouvements de données et des structures de données en traitement d'images[Dul96]

[25]la fonction $sgn(x)$ est définit comme le signe de $x$ par :

$$sgn(x) = \begin{cases} -1 & \text{si} \quad x < 0 \\ +1 & \text{sinon} \end{cases}$$

**Mise à jour des moyennes multiples**
Pour chaque trame $t$,
Pour chaque $i$, $i \in [0, K]$,
Si $t$ est un multiple de $\alpha_i$ :
$m_t^i = m_{t-1}^i + sgn(m_t^{i-1} - m_{t-1}^i)$
**Mise à jour des variances multiples**
Pour chaque trame $t$,
Pour chaque $i$, $i \in [0, K]$,
$\delta_t^i = |I_t - m_t^i|$
if $\delta_t^i \neq 0$ :
$v_t^i = v_{t-1}^i + sgn(N \times \delta_t^i - v_{t-1}^i)$
**Calcul principal**
Pour chaque trame $t$,

$$M_t = \frac{\sum_{i \in [0,K]} \frac{\alpha_i m_t^i}{v_t^i}}{\sum_{i \in [0,K]} \frac{\alpha_i}{v_t^i}}$$

TAB. 3.14 – L'estimation Σ-Δ des fonds multimodaux.

"meilleure confiance" $M_t$, comme montré dans la dernière ligne du tableau. Le principe de ces coefficients de confiance liés à chaque fond $m_t^i$ est de donner plus de poids aux fond à long terme, afin de favoriser les valeurs les plus fréquentes sur de longues périodes, et en même temps, le poids est abaissé selon la variance, afin de permettre l'adaptation à un fond changeant.

La Figure 3.18 montre une application de l'estimation Σ-Δ multipériodique sur une scène complexe. Ceci correspond à une séquence prise dans une partie très fréquentée du Jardin du Luxembourg, à Paris. La scène n'est jamais vide, avec un bon nombre de personnes s'arrêtant pendant des périodes plus ou moins longues. Dans cet exemple, nous avons pris $K = 3$, $\alpha_1 = 1$, $\alpha_2 = 8$, $\alpha_3 = 16$. Ce principe augmente clairement la robustesse de la détection en ce qui concerne la gamme des modèles de mouvement, alors qu'il garde de bonnes capacités d'adaptation aux changements de conditions. Par exemple sur la Figure : à la trame (a), la meilleure confiance est globalement donnée à la moyenne à long terme. A la trame (b), la caméra a été déplacée juste avant, et, dans la partie haute de l'image, les meilleurs résultats sont attribués à la moyenne à court terme, alors que pour la partie basse de l'image la confiance reste haute pour le long terme, ce qui évite l'apparition des deux personnes arrêtées dans le fond final.

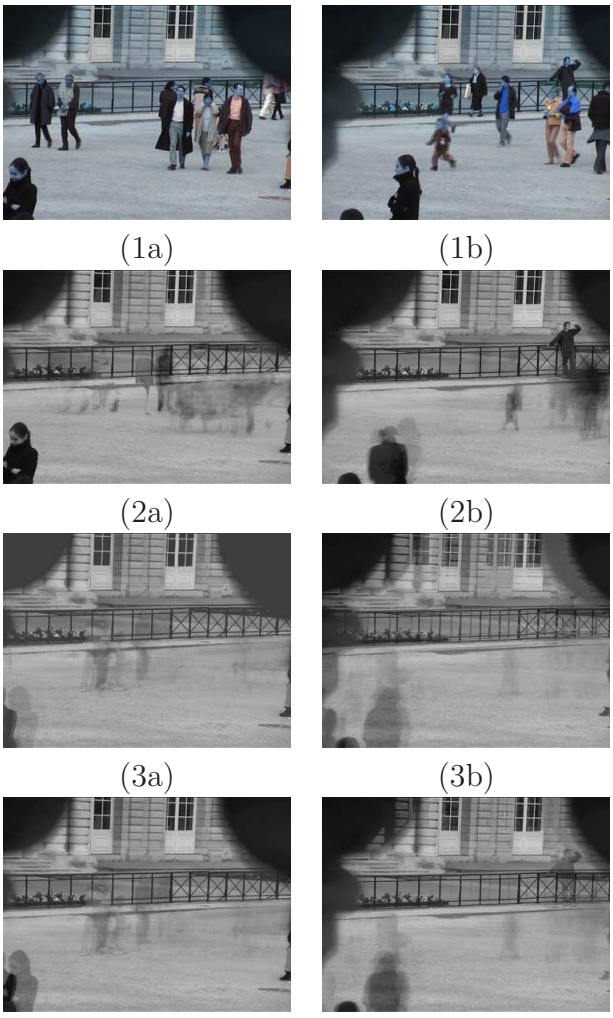



FIG. 3.18 – Estimation Σ-Δ multimodale du fond calculée sur la scène complexe (séquence "Luxembourg"), pour deux instants (a) $t = 1136$ (b) $t = 2896$. (1) Image originale $I_t$. (2) Fond à court terme $m_t^1$ ($\alpha_1 = 1$) (3) Fond à long terme $m_t^3$ ($\alpha_3 = 16$) (4) Fond de meilleur confiance (avec rebouclage) $M_t$.

## 3.8 Conclusion et discussion

Nous avons présenté trois exemples de calcul de détection du mouvement sur système de vision à base de rétine numérique, tenant compte de la quantité limitée de mémoire disponible ainsi que des contraintes du calcul cellulaire.

La Figure 3.19 illustre la comparaison entre les trois algorithmes présentés et portés sur le système rétinien durant notre travail de thèse. L'intérêt de cette figure est de souligner les différences entre la pente unitaire de la moyenne $\Sigma$-$\Delta$ (Figure 3.19(2)) et la décroissance exponentielle de la moyenne récursive (Figure 3.19(1)). On remarque que la moyenne $\Sigma$-$\Delta$ est ainsi moins sujette aux bruits du fait de prise en compte plus progessif des changements du signal. Les erosion et dilatation morphologique oublieuse collent plus au signal et ont donc au contraire du filtre précédent un comportement très adaptatif.

La Figure 3.19(3) montre les résultats de la méthode employant les filtres morphologiques oublieux pour un pixel particulier correspondant à une zone de passage d'un objet en mouvement.

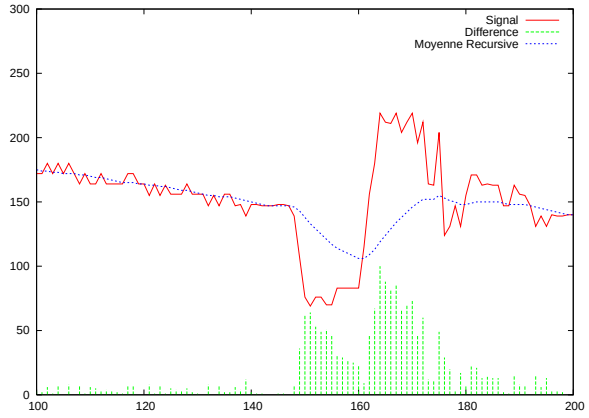
La première méthode présentée, basée sur le calcul d'une moyenne récursive, est peu coûteuse en temps de calcul et en ressources mémoire. Elle est aussi moins adaptative aux conditions de la scène, moins précise en terme de localisation, et plus sujette aux phénomènes de fantôme.

La seconde est une méthode statistique basée sur l'estimation $\Sigma$-$\Delta$. Elle est plus adaptative localement et temporellement. La localisation est plus précise mais elle est plus sensible aux modifications brutales du fond. Un système de filtrage spatio-temporelle basé sur des reconstructions géodésiques permet d'améliorer les résultats de la détection.

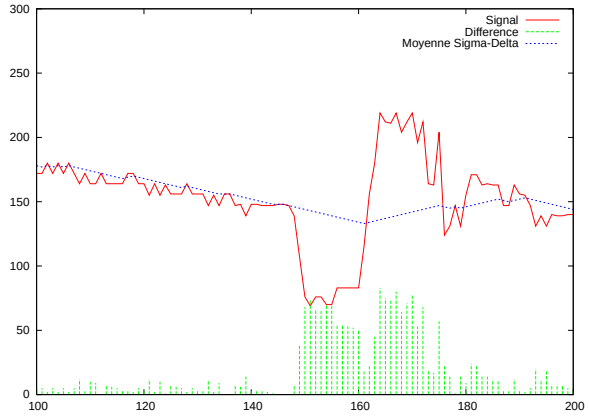
La dernière méthode est une amélioration des techniques récursives basée sur des filtres hybrides. Les opérateurs de morphologie oublieux offrent une plus grande capacité de détection mais sont moins précis dans la localisation des objets détectés.

La Table 3.15 présente un comparatif des trois algorithmes présentés en terme de consommation de mémoire (nombre de registres) et vitesse de calcul (nombre d'instructions à exécuter) sur la rétine numérique.

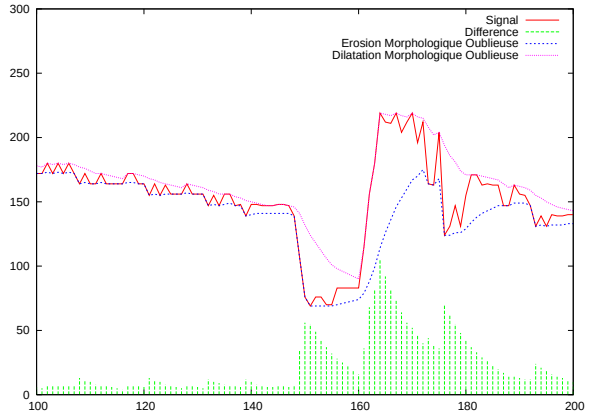
Nous avons ensuite proposé quelques enrichissements en partant des constatations soulignées lors de l'étude des algorithmes. Ces améliorations ont été apportées sur architecture classique et des résultats détaillés ont été présentés.




(1)




(2)




(3)

FIG. 3.19 – Comparaison des trois algorithmes de détection du mouvement présentés pour un pixel appartenant à une zone en mouvement (passage d'un objet mobile). (1) Moyenne récursive. La courbe rouge correspond à la séquence originale, la courbe pointillée bleue au fond et la courbe en tiret vert à la différence au fond (2) Estimation Σ-Δ. La courbe rouge correspond à la séquence originale, la courbe pointillée bleue à la moyenne Σ-Δ et la courbe en tiret vert à la différence à la moyenne (3) Opérateur morphologique oublieux. La courbe rouge correspond à la séquence originale, la courbe violette au maximum oublieux, la courbe pointillée bleue au minimum oublieux et la courbe en tiret vert au gradient morphologique oublieux.

| | **ALGORITHME** | | |
|---|---|---|---|
| | *Acquisition* | | |
| INS | 262 | | |
| REG | 9 | | |
| | *MOYREC* | *MORPHO* | Σ-Δ |
| INS | 122 | 328 | 235 |
| REG | 18 | 36 | 28 |
| | *SEUIL GLOBAL* | | |
| INS | 8 | | |
| REG | 9 | | |
| | *FILTRAGE SPATIAL* | | |
| INS | 66 | | |
| REG | 5 | | |
| | **TOTAL** | | |
| | **MOYREC** | **MORPHO** | Σ-Δ |
| **INS** | **156** | **402** | **309** |
| **REG** | **23** | **41** | **32** |

TAB. 3.15 – Tableau comparatif des trois algorithmes (**MOYREC** pour moyenne récursive, **MORPHO** pour gradient temporel morphologique oublieux et Σ-Δ pour la moyenne Σ-Δ) présentés en terme de temps de calcul (nombre d'instructions (**INS**)) et d'utilisation de l'espace mémoire (nombre de registres utilisés (**REG**)). Le temps mesuré nécessaire pour l'acquisition est de 1,5 ms. Pour une rétine numérique cadencée à 10 Mhz (fréquence d'instructions), le temps de calcul par trame nécessaire pour toute la détection est inférieur à 0,4 ms pour la moyenne récursive, de 1 ms pour le gradient temporel morphologique oublieux et de 0,75 ms pour la moyenne Σ-Δ .

Il semble que le meilleur compromis entre adaptativité et précision de la détection est d'utiliser plusieurs méthodes conjointement ou parallèlement. C'est ce que nous avons montré avec le filtrage utilisé pour les filtres morphologiques oublieux, qui fournit une plus grande robustesse à l'algorithme.

Nous pouvons aussi envisager une sorte d'apprentissage à différents termes (seconde, heure, journée, ...) des caractéristiques de la scène en utilisant des moyennes à différentes fenêtres temporelles construite par décimation de la moyenne récursive (par exemple).

Les deux chapitres suivants introduisent les primitives de calcul spatiales et temporelles nécessaires à la mise en œuvre d'un système d'estimation du mouvement puis présentent les différentes méthodes envisagées sur le système rétinien. En particulier, le chapitre suivant introduit diverses structures (points d'intérêt, squelette, ...) qui sont à la base du suivi d'objets mobiles. Ce chapitre nous permet d'effectuer les transitions entre détection et estimation du mouvement, entre l'utilisation de caméra fixe et de caméra mobile et donc entre la télésurveillance et la robotique.

# Chapitre 4

# Points dominants morphologiques

## 4.1 Introduction

Ce chapitre traite du calcul de primitives qui permettent l'estimation du mouvement, plus particulièrement dans le cadre du suivi d'objets mobiles. Il existe de nombreuses primitives adaptées à ce type de calcul : les primitives 2D (contours, traits, coins, jonctions, ...) ou les primitives 3D (surfaces, crêtes, ...). Nous ne nous sommes intéressés lors de nos recherches qu'aux primitives 2D et plus particulièrement aux points dominants dans l'optique d'effectuer une poursuite de ces points sur plusieurs trames (images) consécutives.

Nous présentons tout d'abord le contexte qui nous a poussé à étudier les structures saillantes spatiales puis nous expliquons les liens existants entre ces structures et les squelettes. Nous effectuons ensuite un bref état de l'art des méthodes de calcul de squelettes binaires, en détaillant particulièrement les techniques qui ont fait l'objet d'un portage sur la rétine. Nous proposons alors une présentation des principales techniques de calcul de points d'intérêts. Enfin, nous présentons un algorithme de calcul de points dominants sur circuit rétinien basé sur l'utilisation des squelettes morphologiques.

## 4.2 Contexte de travail

Dans notre recherche de compacité et d'efficacité du code de calcul numérique en adéquation avec les capacités que nous offrent la rétine numérique programmable, nous cherchons des primitives de calcul nous permettant de réduire l'information utile pour pouvoir appliquer d'autres algorithmes dont ceux de l'estimation du mouvement.

De plus, il existe un lien étroit entre la squelettisation et la notion de courbure spatiale de l'image. En effet, les extrémités des branches des squelettes correspondent à des points de forte courbure du contour des objets. Il a été montré que le calcul de squelette binaire et en particulier l'algorithme MB est adapté aux architectures cellulaires massivement parallèles [Man00]. Une démarche naturelle a donc été d'employer ces squelettes morphologiques afin de fournir un indicateur de la présence de régions de fortes courbures, puis d'intégrer le squelette au sein même d'un opérateur de calcul de points dominants.

## 4.3 Les squelettes

### 4.3.1 Les Squelettes binaires

L'objectif de la squelettisation est de représenter un ensemble avec un minimum d'information, sous une forme qui soit à la fois simple à extraire et commode à manipuler. Le squelette a été introduit par [Blu67] comme l'union des centres des boules maximales inscrites dans l'objet. Il peut être aussi vu comme les résidus de la forme lors d'une succession d'opérations d'érosion. Les points de forte courbure sont situés sur les extrémités du squelette. Plus précisément, la courbure est proportionnelle à la distance entre le contour et le point extrême du squelette.

**Définition 26** *Soit $X$ un ensemble. Une boule $B$ est dite maximale dans $X$ si :*

$$B \subset B' \subset X \Rightarrow B' = B$$

**Propriété 6** *Une boule maximale touche la frontière de $X$ en au moins deux points distincts.*

Dans le cadre des images discrètes, une approche classique pour la squelettisation des objets consiste à amincir l'objet[1] jusqu'à obtenir un ensemble minimal de pixels connectés.

Il existe trois types de squelettes :

1. ceux basés sur la fonction distance[2] ;
2. ceux basés sur une analogie avec les feux de prairies[3] ;
3. et enfin ceux basés sur une suite décroissante d'images homotopes obtenue par amincissement.

En tant que descripteur de forme, un squelette doit satisfaire aux propriétés suivantes :

1. **homotopie** : respect de la topologie. Le squelette doit conserver les relations de connexité : même nombre de composantes connexes, même nombre de trous par composante connexe ;
2. **médialité** : le squelette doit se trouver au milieu de la forme ;
3. **invariance géométrique** : le squelette doit commuter avec la translation, la rotation et l'homothétie ;
4. **immunité au bruit** : le squelette doit être peu sensible à l'ajout ou au retrait de quelques points sur le contour.
5. **épaisseur unité** : le squelette doit être d'épaisseur unité, c'est-à-dire une réunion de courbes ;
6. **réversibilité** : le squelette doit permettre de retrouver la forme originale.

Une méthode classique pour obtenir une approximation du squelette morphologique est d'itérer un processus d'amincissement. En effet, l'algorithme d'amincissement est une méthode qui enlève itérativement des pixels d'une image discrète binaire en préservant certaines propriétés géométriques et topologiques. La relaxation de tels algorithmes conduit au squelette. La popularité des algorithmes d'amincissement provient principalement de la régularité et de la portée des opérations, ce qui les rend faciles à implanter, peu coûteux en termes de ressources mémoire et adaptés aux implémentations parallèles. La Définition 28 présente et formalise les différentes approches déjà évoquées.

**Définition 27 (Amincissement)** *Soit $A$ et $B$ des ensembles de $\mathbb{Z}^2$, $A^c$ est le complémentaire de $A$, $\circledast$ l'opérateur de Transformée en Tout-ou-Rien et $B^n$ la boule de taille $n$ (défini*

[1] On entend par amincir, enlever itérativement des points du contour candidat tout en préservant la topologie et la géométrie

[2] *quench distance* en anglais

[3] *grassfire* en anglais

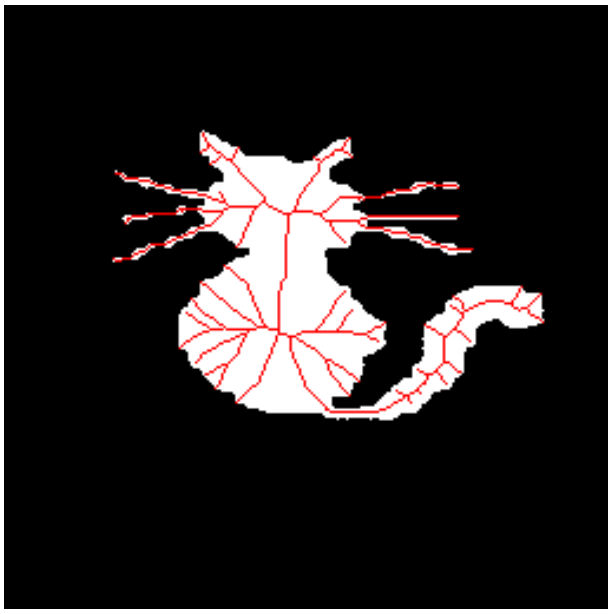


FIG. 4.1 – Exemple de squelette morphologique binaire

*Chapitre 2). L'amincissement*[4] *est défini par :*

$$\begin{aligned} A \otimes B &= A - (A \circledast B) \\ &= A \cap (A \circledast B)^c \\ &= ((...((A \otimes B^1) \otimes B^2)...) \otimes B^n) \end{aligned}$$

Les algorithmes d'amincissement souffrent cependant de deux inconvénients majeurs :

1. il n'existe pas de définition absolue des squelettes digitaux, seulement des définitions algorithmiques. Cela explique la profusion d'algorithmes d'amincissement que l'on trouve dans la littérature [Ser88], [GW92], [GH92], [LER95], ....
2. la métrique des squelettes résultant de l'amincissement dépend de la nature précise des itérations. Ainsi, certaines propriétés désirées comme l'invariance à la rotation ou à l'homothétie sont très difficiles sinon impossibles à obtenir.

[4] *thinning* en anglais

**Définition 28 (Erosion successives)** *Le squelette est défini par :*

$$S(A) = \bigcup_{k=0}^{K} S_k(A)$$

*avec :*

$$S_k(A) = \{(A \ominus kB) - [(A \ominus kB) \circ B]\}$$

$$(A \ominus kB) = \underbrace{((...((A \ominus B) \ominus B) \ominus ...) \ominus B)}_{k\,fois}$$

Dans le cas discret, les boules maximales sont les boules d'une distance discrète donnée. Un point $x$ est centre d'une boule maximale de rayon $r$ dans $X$ si et seulement si il appartient à l'érodé de taille $r$, mais pas à l'ouvert de cet érodé par la boule élémentaire :

$$S_r(X) = \{x \in \mathbb{Z}^3; B(x,r) \text{ est maximale dans } X\}$$

Par conséquent le squelette morphologique est égal à l'union des résidus d'ouverture des érodés successifs de la forme originale :

$$S_K(X) = \bigcup_{r \in \mathbb{N}} \{B_K(x,r) \text{ est maximale dans } X\}$$

Comme l'ensemble des résidus d'ouverture coïncide avec l'ensemble des maxima locaux de la transformée en distance, le squelette morphologique discret est égal aux maxima locaux de la transformée en distance :

$$S_K(X) = \{x \in X; (y \in X \text{ et } \delta_K(x,y) = 1) \Rightarrow (x,r) = (y,r')\}$$

Où $\delta(x,y)$ est la distance Euclidienne de $x$ à $y$.

Mais, contrairement au cas continu, le squelette morphologique discret ne préserve pas nécessairement la topologie de la forme originale. Les algorithmes de squelettisation connexes traitent donc le problème de préservation de la topologie directement dans le cas discret [Man00].

**Théorème de Jordan(rappel)**

**Définition 29 (Proposition de Jordan)** *Dans $\mathbb{R}^2$, toute courbe fermée simple $\partial X$ divise l'espace en deux domaines : un domaine intérieur $X$ et un domaine extérieur $C_E(X)$, chaque domaine étant connexe.*

Dans un maillage carré tout chemin 4-connexe (resp. 8-connexe) simple fermé sépare l'espace en deux composantes 8-connexes (resp. 4-connexes) : l'intérieur et l'extérieur.

Une image binaire est décomposable en un objet et en son complémentaire. Il nest pas possible de conserver la même connexité pour l'objet et son complémentaire car sinon, on voit apparaître des zones de l'image n'appartenant ni à l'objet ni à son complémentaire ou au contraire, appartenant aux deux en même temps (voir Figure 4.2).

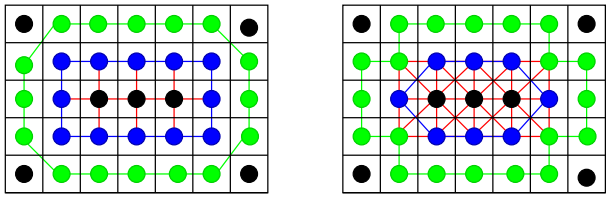


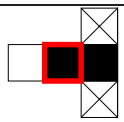
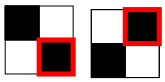
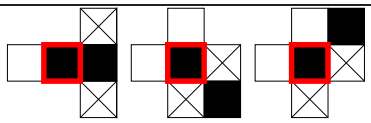
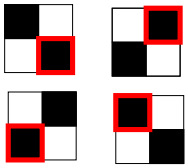
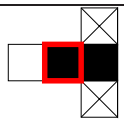
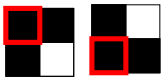
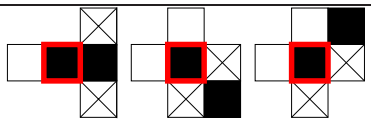
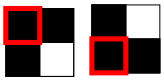
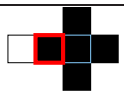
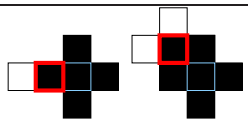
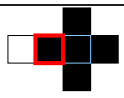
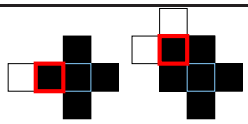

FIG. 4.2 – Le théoréme de Jordan dans la maile carrée est valide si l'on considère des topologies différentes pour l'objet et son complémentaire. (1) frontière intérieure 4-connexe, frontière extérieure 8-connexe ; (2) frontière intérieure 8-connexe, frontière extérieure 4-connexe .

Le théorème de Jordan n'est donc pas valide pour la maille carrée sauf si l'on considère des topologies différentes pour l'objet et son complémentaire. Ainsi, nous devons de la même façon utiliser des connexités différentes lors du calcul du squelette d'un objet et du squelette du complémentaire de l'objet (exosquelette).

### 4.3.2 Les squelettes MB

Dans la pratique, les algorithmes qui calculent des squelettes de formes binaires sont le plus souvent obtenus en appliquant des érosions directionnelles, en conservant les pixels qui introduisent des déconnexions [CT97].

En pratique, le calcul d'un squelette MB[5] consiste à enlever itérativement des pixels correspondant à des configurations dites $\alpha$ tout en conservant ceux se trouvant dans la configuration $\beta$. Les schéma $\alpha$ et $\beta$ peuvent varier en fonction du type de squelette MB (directionnel, parallèle) et de la connexité. De même plusieurs configurations $\alpha$ ou $\beta$ peuvent être utilisées conjointement. pour exprimer par exemple des particularités directionnelles De manière générale, $\alpha$ correspond à des pixels se trouvant sur le bord d'un objet et $\beta$ ceux qui peuvent rompre la connexité, comme des point multiples ou des coins.

| Algorithme | condition pour enlever | condition pour garder |
|---|---|---|
| MBdir1-8 |  |  |
| MBdir2-8 |  |  |
| MBdir1-4 |  |  |
| MBdir2-4 |  |  |
| MBfp1-8 |  |  |
| MBfp2-8 |  |  |
| MBfp1-4 |  |  |
| MBfp2-4 |  |  |

TAB. 4.1 – La famille des squelettes MB.

La Table 4.1 résume les différents algorithmes de la famille des squelettes MB et les

[5]MB pour les créateur du squelette, A. MANZANERA et T. BERNARD

différentes configuration de $\alpha$ et de $\beta$ qui leurs sont associées. A cette famille, nous devons ajouter les squelettes hybrides, qui sont une composition de squelettes directionnels et parallèles [BM99]. Une itération complète de l'algorithme du squelette MB Hybride consiste en une itération d'un algorithme parallèle suivi par quatre itérations d'un algorithme directionnel semi-parallèle dans les quatre directions cardinales successivement (Nord, Est, Sud, Ouest). Une telle combinaison permet une meilleure invariance en rotation. Approximativement, cela produit une branche de squelette pour chaque angle inférieure à $\pi/2$ dans chaque direction. Les structures booléennes de l'algorithme et les détails de son implémentation se trouvent en [MB03].

La Figure 4.3 présente les résultats des différents algorithmes de squelettisation sur le banc algorithmique de la rétine et calculés en temps réel.

### 4.3.3 Les squelettes en niveau de gris

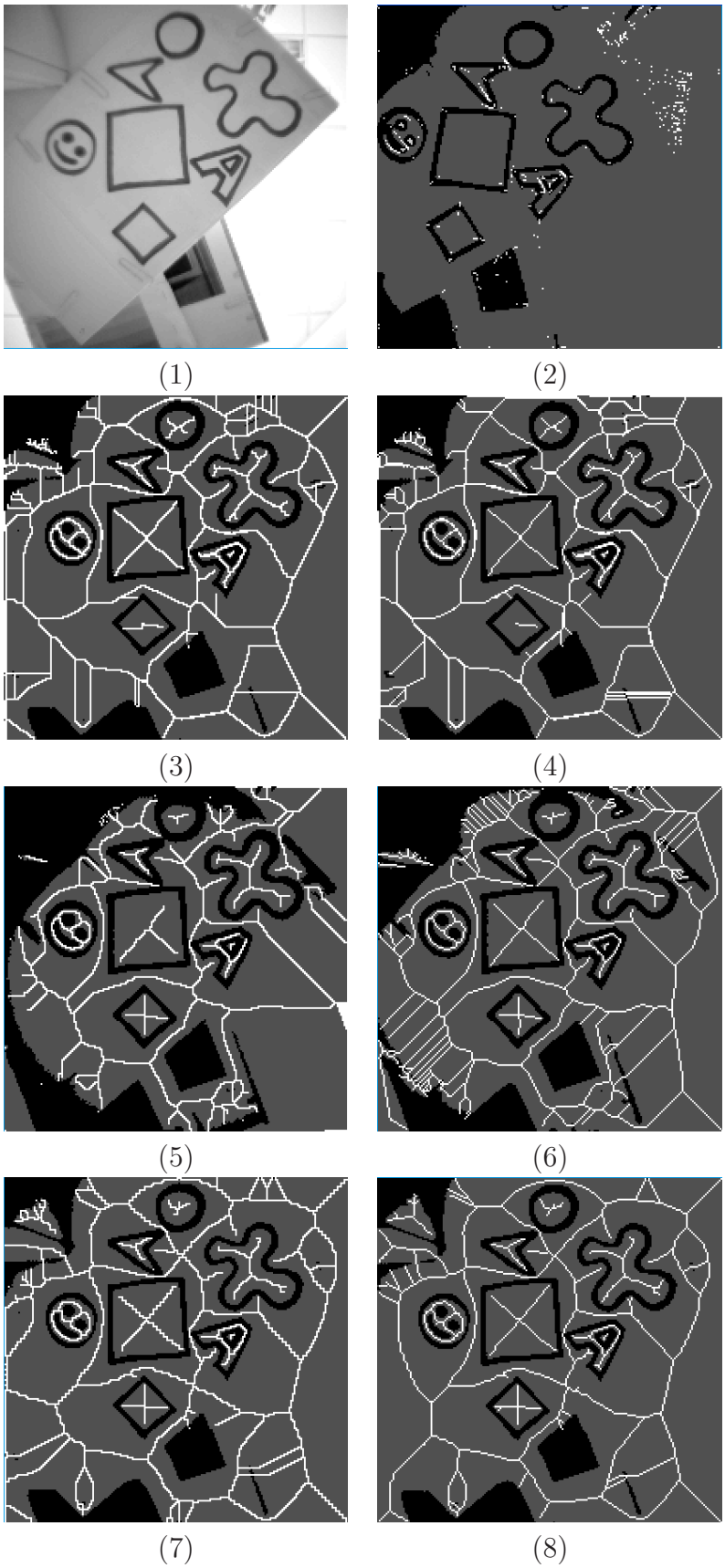
Parmi les travaux majeurs concernant le calcul d'un squelette directement en niveaux de gris nous pouvons citer [Dok00] [Bez01] [CBB01]. Ces travaux sont basés sur des notions topologiques dont le noyau homotopique[6]. Les opérations de squelettisation et d'amincissement sont en effet les applications majeures des calculs topologiques en niveaux de gris. Une plus large présentation du calcul de squelettes en niveau de gris est faite en Annexe B.

#### Squelette MB en niveau de gris

Nous avons adapté le calcul des squelettes MB dans le cadre des squelettes en niveaux de gris. Dans un premier temps nous nous sommes inspirés de l'algorithme de calcul des $\lambda$-squelettes ([Dok00][Bez01]) en se limitant au calcul de noyaux homotopiques. Pour cela, nous utilisons le nombre de connexité que nous avons défini dans le Chapitre 2, Définition 12.

Le principe de notre calcul d'un squelette MB en niveaux de gris est de combiner les aspects topologiques des squelettes MB binaires avec l'utilisation de masques $\alpha$ et $\beta$ tout en s'inspirant des $\lambda$-squelettes et du calcul des noyaux homotopiques.

[6]Il s'agit d'un ensemble de pixels minimal préservant les notions de topologies de l'objet. En niveau de gris, il s'agit de sélectionner des pixels particulier (points destructible) et d'abaisser leurs niveaux de gris jusqu'à une valeur où il n'est plus destructible, et en répétant ce processus jusqu'à stabilité.



(1) (2)

(3) (4)

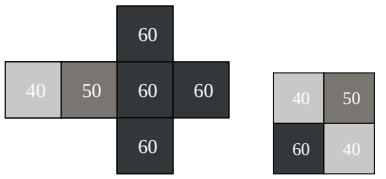
(5) (6)

(7) (8)

FIG. 4.3 – Résultats des algorithmes de squelettisation MB1 sur le banc de test rétinien : (1) Image originale ; (2) Points d'intérêt morphologiques ; (3) MB1 Full-Parallel 4-connexe (4) MB1 Full-Parallel 8-connexe (5) MB1 Directionnel 4-connexe (6) MB1 Directionnel 8-connexe (7) MB1 Hybride 4-connexe (8) MB1 Hybride 8-connexe .

**Répéter** jusqu'à stabilité**{**
**Pour** chaque pixel $p(x, y)$ de $I_{in}$ tel que $(I(x, y-1) < I(x, y))$**{**
$\mathcal{N}_8(x) = \sum_{i=0}^{3}(x_{2i+1} \vee x_{2i+2}) \wedge \overline{x_{2i}}$
**Si** $\mathcal{N}_8(x) = 1$ et $I(x, y)$ n'est pas terminal
**alors {** $max = 0$ ;
**Pour** tous les voisins $q$ de $p$ **{**
**Faire si** $(I(q) < I(p))$ et $(I(q) > max)$ **alors** $max = I(q)$
$I_{out}(p) = max$**}**
**}**
**}**
**}**

TAB. 4.2 – Algorithme de calcul du noyau homotopique en niveaux de gris.




$\alpha_{gris}.Nord$ $\beta_{gris}.SW$

FIG. 4.4 – Exemple de masque conditionnelle utilisés pour les squelette MB en niveau de gris.

Ainsi, afin de passer du calcul de noyaux homotopiques à celui de squelettes MB en niveaux de gris, il nous faut ajouter le calcul des masques conditionnelles $\alpha$ et $\beta$ dans leur version en niveaux de gris. La Table 4.3 présente l'algorithme de calcul du squelette directionnel MB en niveaux de gris en prenant les masques $\alpha$ pour l'itération Nord ainsi que l'un des deux masques pour $\beta$ présentés Figure 4.4.

**Répéter** jusqu'à stabilité**{**
  **Pour** chaque direction $N$, $E$, $S$ et $W$**{**
    **Pour** chaque pixel $p(x,y)$ de $I_{in}$**{**
      Calculer $\alpha$ et $\beta$ ;
      **Si** $\alpha \wedge \bar{\beta}$
      **alors** $p(x,y) = max\{p(i,j) | p(i,j) < p(x,y)\}$ ;
    **}**
  **}**
**}**

TAB. 4.3 – Algorithme de calcul du squelette directionnel MB en niveaux de gris.

Une dernière étape consiste à calculer le $\lambda$-squelette MB en niveaux de gris en intégrant le paramètre $\lambda$ dans le calcul du squelette MB en niveaux de gris. Cela revient à introduire un terme d'échelle de niveaux représentant le seuil minimal à prendre en compte pour qu'une différence soit significative. Contrairement au cas des squelettes simples, le paramètre $\lambda$ dans les squelettes MB ne rend pas le squelette moins bruité mais limite la profondeur de précision du squelette.

D'un point de vue visuel, les squelettes en niveaux de gris remplissent leur rôle et l'on retrouve bien certaines des propriétés évoquées pour les squelettes binaires, dont la médialité, l'invariance en translation, homothétie et certaines rotations . Cependant, ces squelettes ont beaucoup de branches non significatives que l'on peut difficilement limiter même avec la mise en place d'un paramètre de contrôle comme le paramètre $\lambda$. Ils sont beaucoup plus bruités et la manière dont ils sont construits provoque un flou important le long des frontières entre deux régions adjacentes au contraire des squelettes binaires.

L'implémentation machine de ces squelettes est difficile et le temps de calcul nécessaire pour leur stabilité est important, empêchant d'envisager leur portage sur le circuit rétinien dans l'état. De plus, nous avons envisagé de les utiliser dans le cadre de calcul rapide de points d'intérêt directement en niveau de gris en les utilisant afin d'estimer les points de forte courbure. Or, en réalité les branches du squelette MB en niveaux de gris représentent à la fois des points de fortes courbures d'un point de vue spatial mais aussi des régions de non homogénéité à l'intérieur de régions plates. Avec ces techniques on obtient une multitudes de points le long des frontières de ces objets avec une dispersion proche d'une

gaussienne à l'intérieur des régions. Ils ne sont donc pas exploitables dans le cadre de calcul de fonctions d'intérêt directement en niveaux de gris. Les résultats obtenus sont certes décevants dans l'optique d'opérer une estimation du mouvement mais pourront servir pour des travaux ultérieurs notamment sur des questions de calcul par ensemble de niveaux ou encore sur des questions de voisinages tels que la segmentation.

### 4.3.4 Conclusion

La Figure 4.5 présente une série d'exemples de squelettes binaires et en niveaux de gris sur un détail de l'image de test "Lenna ". On peut voir la similitude entre le squelette binaire morphologique de l'image seuillée avec le squelette hybride en niveaux de gris. On remarque les différences induites par les diverses définitions de squelettes en niveaux de gris, squelettes topologiques, squelettes MB et squelettes hybrides[7].

Les squelettes permettent de réduire considérablement l'information contenue dans les images permettant d'envisager le suivi d'objets en mouvement. Dans cette optique, plus le nombre de points à suivre est réduit plus le calcul sera aisé. Nous n'avons ainsi pas besoin de disposer du squelette dans son intégralité, seuls quelques points (les extrêmités des branches des squelettes) nous suffisent.

Or, il a été montré ([CT97] [MB03]) que les extrêmités des branches des squelettes binaires, notamment MB, offrent une bonne approximation de la courbure binaire locale. En raisonnant sur les ensemble de niveaux, finalement, la meilleure implantation en niveau de gris du calcul de la courbure revient à effectuer le calcul des extrêmités des branches d'un squelette binaire pour chaque niveau de gris.

C'est pourquoi nous présentons dans la section suivante un nouvel opérateur de calcul de points dominants morphologiques utilisant les squelettes morphologiques MB Hybrides, après avoir effectué une présentation des techniques classiques de calcul des points d'intérêt.

[7]Les squelettes en niveau de gris font l'objet d'une partie des annexes, ceux-ci ne faisant pas partie directement du travail produit dans le cadre de la détection et l'estimation du mouvement

(1) (2) (3)

(4) (5) (6)

FIG. 4.5 – Exemple de squelette binaire et en niveau de gris sur un détail de l'image de test "Lenna" :
(1) Image originale
(2) Image seuillée
(3) Squelette morphologique binaire de l'image seuillée
(4) Squelette en niveau de gris
(5) Squelette morphologique en niveau de gris
(6) Squelette morphologique hybride en niveau de gris

## 4.4 Points d'intérêt

### 4.4.1 État de l'art

Les points d'intérêt[8] correspondent à des changements bidimensionnels locaux importants de la fonction d'intensité, obtenus :
- soit par le calcul (étude locale de l'intensité) ;
- soit par segmentation préalable ;
- soit par une étude multirésolution.

Ce sont des caractéristiques bas niveau. Le signal contient plus d'information en ces points qu'en des points correspondant à des changements unidimensionnels du signal ou à des régions homogènes [Lou02].

Ces points sont par exemple les coins (en L avec un angle plus ou moins obtus) et les jonctions (en T ou bien en Y) mais aussi correspondent aux endroits de l'image où la texture varie de façon significative. De tels points sont très utiles dans de nombreuses applications comme l'indexation vidéo ou encore la reconstruction 3D en vision artificielle. La Figure 4.6 illustre les différents types de points d'intérêt rencontrés lors de notre étude.

Globalement, nous pouvons ranger les algorithmes de détection des points d'intérêt en trois catégories :

Les approches du premier groupe font d'abord une **extraction de contours** puis suivent ensuite les points de ces contours à la recherche de points de courbure maximale. Ils peuvent aussi effectuer une approximation polygonale sur le contour et chercher les intersections.

La seconde approche consiste à travailler directement sur l'image en niveaux de gris. Cette technique est basée à la fois sur des **opérateurs locaux** empiriques et sur des mesures de gradients ou de courbure de surface.

Les approches de la dernière catégorie sont basées sur des outils de **morphologie mathématique**. Ce sont des techniques récentes qui utilisent les propriétés d'approximation des contours à l'aide des résidus d'ouverture morphologique ou de détection de formes particulières par Transformation en Tout ou Rien (TTR).

[8] *interest point* ou *key point* ou *corner point* en anglais

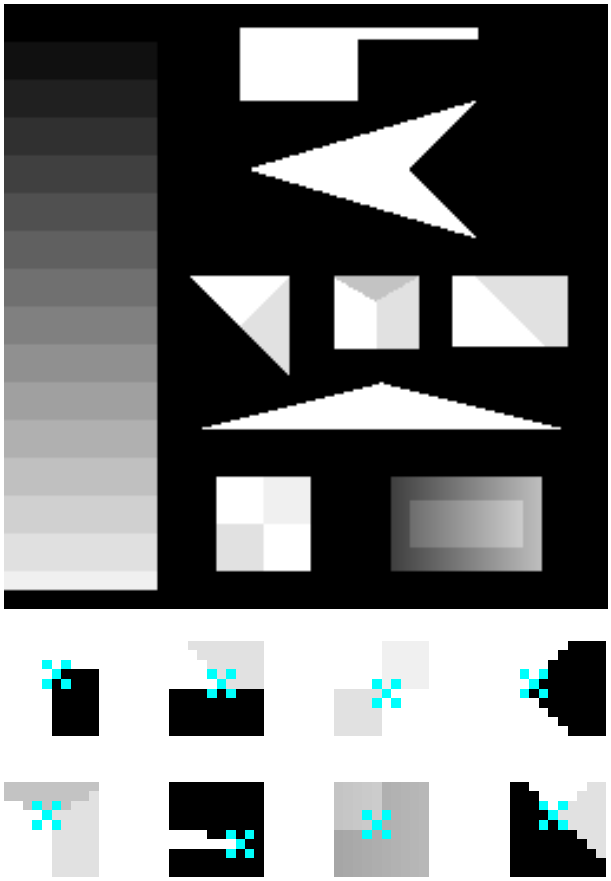


FIG. 4.6 – Exemples de points d'intérêt issus de l'image de test de Smith.

**Méthodes basées sur les contours**

Le principe des méthodes basées sur les contours est soit de rechercher les points de courbure maximale le long des chaînes de contours soit d'effectuer une approximation polygonale en vue d'en déduire des points particuliers. De telles méthodes existent depuis longtemps, seules les plus pertinentes pour notre contexte sont présentées.

Asada et Brady [AB86] extraient des points d'intérêt pour des objets 2D à partir de courbes planes décrivant les contours. Ils constatent que les courbes planes ont des caractéristiques significatives : les changements de courbure. Afin de détecter ces changements de manière robuste, l'algorithme est intégré dans un cadre multi-échelle. Une approche similaire a été proposée en utilisant les points d'inflexion des courbes planes.

Horaud [HV90] recherche des groupements dans une image de contours pour établir une représentation intermédiaire reposant sur la structuration de segments extraits de l'image. L'intersection de ces segments donne les points d'intérêt.

Deriche et Giraudon [DG93] utilisent à la fois l'approche "intensité" et l'approche "contour" pour détecter les coins. Ils calculent les maxima d'un opérateur à deux échelles différentes (lissages) puis prennent l'intersection de la droite joignant les points ainsi calculés avec celle des points correspondant aux passages par zéro du laplacien de l'image.

D'autres approches basées contours ont été proposées que nous pouvons résumer ainsi :
- une méthode approxime les contours avec des B-splines : les points d'intérêt sont des maxima de courbure qui ont été calculés par les coefficients de ces B-splines [MY87].
- une approche est basée sur une décomposition de courbes discrètes bruitées en un nombre minimal de sections concaves et convexes. La détection des points d'intérêt est basée sur les propriétés d'appariements de ces sections [PD94].
- une approche détecte d'abord les crêtes et les creux dans l'image [AD91]. Les points d'intérêt sont les points de haut de courbure le long des crêtes, des creux ou des points d'intersection [SWG97].
- une approche décrit un détecteur de points d'intérêt basé sur l'appartenance à deux ensembles. Le premier de ces ensembles est composé de T-jonctions extraits des contours et le second est obtenu en utilisant un cadre multi-échelles : initialement, les points d'intérêt sont des maxima de courbure des contours et sont ensuite "pistés" localement jusqu'au niveau le plus élevé. Finalement, on ne retient que les points présent sur l'ensemble de l'espace d'échelle et dans le premier ensemble [MS98].

**Méthodes basées sur le signal**

Les méthodes basées sur le signal détectent directement les points d'intérêt à partir de l'information locale de niveau de gris. Il existe de nombreuses techniques issues de ces méthodes, nous présentons ici les plus importantes.

Le détecteur de Beaudet [Bea78] utilise les dérivées secondes du signal pour calculer une mesure appelée "$DET$", :

$$DET = I_{xx}I_{yy} - I_{xy}^2$$

où $I(x, y)$ représente l'intensité de l'image. DET est le déterminant de la matrice du Hessien et est lié à la courbure Gaussienne du signal. Cette mesure est invariante en rotation si les dérivées sont calculées à partir de convolution avec des dérivées de noyaux gaussiens.

Moravec [Mor79] a proposé un détecteur basé sur la fonction d'auto-corrélation du signal :

$$MO(x, y) = \frac{1}{8} \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} |I(i, j) - I(x, y)|$$

Cette fonction mesure les différences entre un pixel et ses huits voisins directs. Lorsque le minimum de ces différences est supérieur à un seuil, il indique alors la présence d'un point d'intérêt.

Dans une autre approche, on approxime la fonction image I au voisinage du pixel $(x, y)$ par un polynôme bi-cubique ayant comme coefficient $c_k$ :

$$\begin{aligned} I(x, y) = c_1 + c_2x + c_3y + c_4x^2 + c_5xy + c_6y^2 \\ + c_7x^3 + c_8x^2y + c_9xy^2 + c_{10}y^3 \end{aligned}$$

Cette équation a donnée naissance à plusieurs détecteurs :

$$ZH(x, y) = \frac{-(c_2^2c_6 - 2c_2c_3c_5 + c_3^2c_4)}{(c_2^2c_3^2)^{\frac{3}{2}}} \text{ (Zuniga et Haralick (1983))}$$

et

$$KR(x, y) = \frac{-(c_2^2c_6 - 2c_2c_3c_5 + c_3^2c_4)}{c_2^2c_3^2} \text{ (Kitchen et Rosenfeld (1982))}$$

Autour d'un coin la courbure gaussienne change de signe et possède un maximum positif et un minimum négatif. Il est donc possible de localiser un point d'intérêt sur la

ligne joignant ce minimum et ce maximum, notamment à l'endroit où la pente du signal est maximale, c'est-à-dire où la courbure s'annule.

Le détecteur de Harris fait partie de cette famille de détecteur, et utilise une auto-corrélation de l'image [HS88]. Un lissage gaussien (d'écart-type $\sigma$) est effectué avant de calculer les dérivées du signal. La matrice $A$ est calculée en chaque pixel :

$$A = e^{-\frac{x^2+y^2}{2\sigma^2}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

où $I_x$ est la dérivé du premier ordre selon $x$. Les dérivées sont calculées avec le masque de convolution [-2 -1 0 1 2] tandis que dans sa version améliorée, elle sont calculées à l'aide de la fonction Gaussienne. Les vecteurs propres de cette matrice correspondent aux principales courbures de la fonction d'auto-corrélation [SM99]. Si ces deux courbures sont grandes, ceci indique la présence d'un point d'intérêt. Toutefois, pour ne pas extraire les valeurs propres, une mesure est proposée reposant sur le déterminant et la trace de la matrice. Cette mesure, appelée fonction de Harris et définie par $F_H = det(A) - \alpha tr(A)^2$, est supérieure à zéro dans le cas d'un coin et inférieure à zéro pour un contour. Ce détecteur est le détecteur de coin le plus utilisé dans de nombreuses applications et a été évalué comme étant le meilleur détecteur de coins par différentes études comparatives dont [SMB98]. Cependant, de nombreuses implémentations de ce détecteur existent, car cinq paramètres doivent être choisis lors de son utilisation :

1. le filtre dérivatif,
2. le filtre gaussien ($\sigma$),
3. le paramètre du détecteur ($\alpha$),
4. le voisinage de l'extraction des maxima locaux, et
5. le seuillage final.

Mikolajczyk et Schmid [MS01] proposent un détecteur qui combine le détecteur de Harris avec l'échelle d'invariance du **détecteur de Lindeberg**[9] [Lin98]. Dans un premier temps, une représentation multi-échelle de la fonction de Harris $F_H$ est construite. A chaque niveau de l'**espace d'échelle**, un maximum local de la fonction de Harris est recherché par :

$$F_H(\vec{x}, \sigma_n) > F_H(\vec{x}_w, \sigma_n), \quad \forall \vec{x}_w \in W$$

où $W$ est le voisinage en 8-connexité du point pour $\vec{x}$. Ils obtiennent bien entendu un opérateur plus performant mais avec un coût calculatoire bien plus important, comme celà est le cas avec l'utilisation d'opérateurs multi-échelles.

[9]Lindeberg définit un extremum normalisé de l'espace d'échelle par un point qui est simultanément un extremum local en ce qui concerne le domaine spatial et le paramètre d'échelle. Tant que l'espace d'échelle et le domaine spatial sont discrets, la détection de points d'intérêt est mise en application en recherchant un extremum dans les trois dimensions (x, y, et échelle).

Enfin, bien d'autres approches ont été proposées citons simplement ces dernières :
- les approches "*tracking*" qui partent du principe qu'un bon point d'intérêt est celui qu'il est facile de suivre : on démontre que de tels points sont présents si les valeurs propres de la matrice d'auto-corrélation $A$ sont significatifs [TK01].
- les approches inspirées des mécanismes neurobiologiques qui consistent à convoluer l'image avec des filtres directionnels pairs et impairs. Ces filtres sont des fonctions sinusoïdales sur une enveloppe gaussienne de moyenne nulle proche des **filtres de Gabor** (Chapitre 5 Partie 5.2.3) [Ha90].
- les approches qui utilisent un masque circulaire centré sur le pixel afin de comparer l'intensité de chaque pixel au centre et d'identifier certaines régions pouvant contenir un point "intéressant" (par exemple, SUSAN-2D[10] [SB97]).
- les approches qui consistent à mesurer d'abord la direction du contour localement et font ensuite une différence d'image le long du contour. Une connaissance des caractéristiques du bruit est utilisée pour déterminer où cette différence est suffisante pour localiser les points d'intérêt [CVK93].
- une approche en deux étapes :
  1. Les points sont d'abord détectés sur une fenêtre optimale en utilisant la matrice $A$ (voir ci-dessus). Cette détection provoque systématiquement des erreurs de localisation, par exemple dans le cas de "L-corners".
  2. Une seconde étape basée sur un calcul différentiel des frontières d'intersection améliore le taux de localisation [FG87].
- enfin, les approches qui généralisent les points d'intérêt aux espace d'échelle en définissant un extremum normalisé d'une entité différentielle comme un point de l'espace d'échelle qui est simultanément un extremum local en ce qui concerne le domaine spatial et le paramètre d'échelle. Puisque l'espace d'échelle et le domaine spatial sont discrets, la détection de points d'intérêt est implémentée en cherchant un extremum dans les trois dimensions (x,y et échelle). Encore une fois, les performances de ces opérateurs sont d'autant plus grandes que le coût est élevé. Un compromis temps de calcul et qualité des résultats est donc souvent fait avec l'utilisation des espaces déchelles qui se traduit souvent par un appauvrissement des espaces d'échelles utilisées.

### Méthodes basées sur les outils de la morphologie mathématique

Peu de travaux ont porté sur l'élaboration d'un détecteur de points d'intérêt à base d'outils de morphologie mathématique. [ZZ95] détecte des points coins convexes sur des

[10]SUSAN pour *Smallest Univalue Segment Assimilation Nucleus*

contours fermés dans des images binaires. Ce détecteur est basé sur le calcul de résidus successifs d'ouvertures morphologiques.

**Définition 30** *Soit $X$ une image, $X^c$ son complémentaire et $D(n)$ l'élément structurant de taille $n$. Le détecteur de points d'intérêt morphologiques*[11] *introduit par [ZZ95] est défini par :*

$$[(X - (X \circ D(n))) \cup (X^c - (X^c \circ D(n)))] \cap (X - (X \ominus D(n)))$$

Nous pouvons caractériser ce détecteur par : *"les points appartenant au résidu de l'image ou au résidu du complémentaire de l'image et au contour binaire de l'image"*. Ce détecteur est sensible à la taille de l'élément structurant choisie ainsi qu'à sa forme. Cependant, les propriétés de lissage du détecteur, grâce notamment à l'utilisation de Top-hat (Chapitre 2, Définition 51) lui confèrent une bonne résistance au bruit. Les auteurs ont pris en compte les particularités d'un calcul sur une architecture SIMD massivement parallèle mais ne fournissent pas de résultats ni qualitatifs, ni quantitatifs.

L'approche proposée par [Lag98] est basée sur une variante de l'opérateur morphologique de fermeture pour détecter des L-corners verticaux et horizontaux et des L-corners diagonaux. Enfin, plus récemment [DG04] propose de détecter les pics de la courbure locale à l'aide des outils issus de la morphologie mathématique.

#### Conclusion

Pour une revue plus exhaustive des travaux réalisés dans le domaine des points d'intérêt, le lecteur peut se reporter aux travaux de [SMB98], [TC89], [SM99] et [BJ99]. Dans la suite du document nous supposons que les points d'intérêt sont des points de forte courbure de la fonction d'intensité locale. De tels points sont abondamment présentés et illustrés dans la section suivante.

### 4.4.2 Points dominants morphologiques

#### Introduction et préliminaires

Notre approche consiste à définir une fonction d'intérêt basée sur une mesure locale qualitative de la courbure à travers les ensembles de niveaux de l'image en niveaux de

[11] nommé *filling* par les auteurs

gris. Nous utilisons un algorithme de squelettisation afin d'approximer le calcul des points de forte courbure.

Cet algorithme repose sur l'emploi d'outils morphologiques particulièrement bien adaptés aux calculs sur machine SIMD cellulaire. Nous présentons l'implémentation sur rétine programmable qui permet un calcul parallèle efficace basé sur les ensembles de niveaux.


FIG. 4.7 – Exemple du calcul des points dominants morphologiques sur l'image de test issue de la séquence "movi" (trame numéro 9).

La Figure 4.7 montre un résultat du calcul des points dominants morphologiques sur l'image de test issue de la séquence "movi" provenant de la bibliothèque d'images de l'INRIA[12]. Il s'agit en fait d'une superposition de la fonction d'intérêt seuillée ($\delta = 13$) et de l'image originale. Les points d'intérêt ont subi une dilatation par une croix ($\times$) pour

[12]Institut National de Recherche en Informatique

pouvoir être localisés rapidement de manière visuelle.

### Courbure et fonction d'intérêt

Nous posons dans cette section les bases préliminaires au calcul des points dominants. Nous apportons ainsi les définitions de la courbure que nous utilisons dans la suite de notre exposé.

**Définition 31** *Dans un espace Euclidien, la courbure est définie par le changement de signe instantané des dérivées de la fonction d'intensité[AB86]. Nous supposons que $y(x)$ représente une fonction de courbure, et $R_p$ est la courbure au point p, alors $R_p$ est définie par les dérivées de y :*

$$R_p = \frac{\frac{\partial^2 y}{\partial x^2}}{[1 + (\frac{\partial y}{\partial x})^2]^{3/2}}$$

*La courbure est proportionnelle au rayon du cercle tangent inscrit maximal à la courbe au point p .*

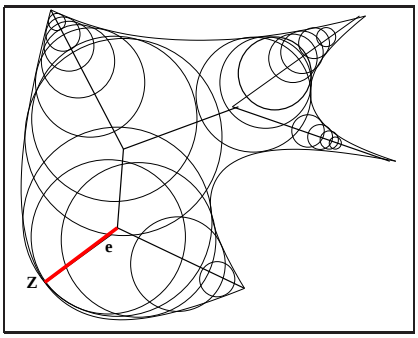



FIG. 4.8 – Mesure de la courbure locale. Le rayon du cercle inscrit de rayon $e$ est fonction de la courbure locale au point $Z$. Plusieurs cercles inscrits à la forme ont été tracés afin d'illustrer le lien entre réunion de cercles maximum et squelette que nous utilisons dans la suite du chapitre (Partie 4.3).

**Définition 32 (courbure de la ligne de gradient orthogonale aux isophotes)** *La courbure aux isophotes est définie par :*
$g_\perp = (I_{xx} - I_{yy})I_x I_y - I_{xy}(I_x^2 - I_y^2) = I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_x I_y$

Pour des images binaires, une fonction d'intérêt naturelle est fournie par la courbure locale (Définition 33).

**Définition 33** *Soit $\overset{\circ}{X}$ le contour de $X$. La fonction d'intérêt $\varphi_X$ associée à l'image binaire $X$ est alors définie par :*

$$\varphi_X(p) = \begin{cases} R_p \quad si \quad p \in \overset{\circ}{X} \\ 0 \quad sinon \end{cases}$$

*où $R_p$ représente la courbure locale (voir Définition 31).*

**Définition 34** *D'après les propriétés données par les ensembles de niveau , nous pouvons maintenant définir la fonction d'intérêt $\phi$ pour une image $I$ en niveaux de gris par la somme des fonctions d'intérêt $\varphi_{I_t}$ à travers les ensembles de niveau $I_t$ de l'image.*

$$\begin{array}{rccc} \phi: & E & \rightarrow & \mathbb{N} \\ & p & \mapsto & \displaystyle\sum_{t \in K} \varphi_{I_t}(p) \end{array}$$

Nous remarquons qu'en remplaçant dans la Définition 33 $R_p$ par une valeur constante, nous obtenons le gradient morphologique (voir Définition 49). Ainsi, la fonction $\phi$ donne une mesure de l'intérêt des images en niveaux de gris en pondérant le contraste local par la courbure.

Notre problème est la recherche de points dominants, c'est à dire de points du contour de forte courbure. Ce sont des points où le cercle inscrit à la forme converge vers un cercle de rayon infiniment petit. Or, nous pouvons définir justement un squelette morphologique binaire comme la réunion du centre des boules maximales. Ainsi, il existe une analogie entre la boule de l'extrémité d'une branche du squelette et le centre du cercle inscrit au point dominant du contour. En réalité le squelette n'est qu'une approximation du centre des cercles maximaux inscrits au contour de la forme.

Le choix d'utiliser le squelette présente plusieurs intérêts. En premier lieu, il permet d'obtenir les points dominants convexes et concaves en effectuant le squelette sur le complémentaire. Ensuite, le squelette est adapté à un calcul par ensemble de niveaux car il a été conçu pour des images binaires.

On peut remarquer que nous parlons de contours par ensembles de niveau. Il ne s'agit pas nécessairement des contours des objets contenus dans la scène. En effet, une singularité

locale de la fonction d'intérêt entraînant une perturbation dans le calcul du squelette peut être à l'origine pour le pixel d'une réponse pour un niveau. Cependant, comme nous intégrons les réponses niveau par niveau, le point sera considéré comme "intéressant" s'il a une réponse sur un certain nombre de niveaux ; déterminé par le niveau de seuillage de la fonction d'intérêt. Ainsi, nous plaçons nos points dominants majoritairement sur le contour des objets mais aussi dans les zones où la texture des objets présente elle aussi une courbure importante.

La section suivante présente en détail l'implantation de notre algorithme sur la rétine artificielle numérique programmable.

### Implémentation

Nous disposons de deux techniques nous permettant d'effectuer un calcul par ensembles de niveau : effectuer un seuillage niveau par niveau puis d'agréger les résultats selon certains conditions en fonction des opérateurs utilisés ; ou effectuer le calcul au cours de l'acquisition de l'image par le circuit et de propager les résultats intermédiaire au cours du temps (de la décharge de la photodiode, voir Chapitre 2). La seconde technique, bien que plus efficace en terme d'utilisation du circuit et d'optimisation de temps de calcul, est plus difficile à mettre en place d'un point de vue algorithmique. Les résultats produits étant équivalents nous avons utilisé la première solution, tout du moins, dans un premier temps et lorsque l'algorithmie, par le temps disponible entre deux acquisition, nous le permettait.

La Table 4.4 montre le déroulement de l'algorithme de calcul des points dominants morphologiques sur un système à base de rétine numérique. Le résultat de ce calcul nous donne une fonction d'intérêt dont la plage de valeurs dépend du nombre de seuillages effectués et de la dynamique de l'image. Une image codée sur 8 bits nous donnera une fonction d'intérêt codée sur 8 bits. Ainsi, une manière immédiate de réduire le temps de calcul nécessaire à la détection des points dominants est de "décimer" le nombre de seuillage et de coder la fonction d'intérêt sur un nombre de bits moins importants. On peut justifier ce principe par le fait qu'il n'est pas rare qu'un point de forte courbure pour un niveaux de gris $x$, le soit aussi pour le niveau $x+1$ ou $x+2$, voir $2x$.

Cependant, ce "raccourci" n'a pas été employé dans la mesure où le temps de calcul de notre algorithme est raisonnable quant à son exploitation dans un système de détection du mouvement. Par contre, il nous a permis de trouver une méthode plus fine de seuillage de la fonction d'intérêt. En effet, un simple seuillage de la fonction $F$ obtenue était suffisant la plupart du temps mais pouvait parfois conduire à un nombre soit trop important, soit pas

assez suffisant de points détectés. Nous avons expérimenté plusieurs techniques, et c'est celle des maxima locaux de fonction d'intérêt qui nous a donné les meilleurs résultats.

**Pour chaque ensemble de niveau** $n$ **{**
1. Acquisition du niveau de gris $I_n$ par seuillage ;
2. Copie et inversion en $I_n$ et $I_n^c = NOT(I_n)$ ;
3. Une itération d'amincissement des images $S(I_n)$ (squelette) et $S(I_n^c)$ (exosquelette) ;
4. Calcul des points extrêmes $ep(S(I_n))$ et $ep(S(I_n^c))$ ;
5. OU logique entre les deux images $F_n = ep(S(I_n))\ OR\ ep(S(I_n^c))$ ;
6. Accumulation de la somme $F = F + F_n$

**}**

TAB. 4.4 – Algorithme de calcul des points d'intérêts morphologiques.

L'algorithme de squelette utilisé dans notre détecteur de points dominants est l'algorithme de squelettisation hybride MB1. Les avantages de cet algorithme sont caractérisés par :

1. une certaine **invariance en rotation** ;
2. une production réduite de **branches non significatives** ;
3. un **coût de calcul faible** ;
4. l'existence d'une version 8-connectée et une version 4-connectée avec les **mêmes propriétés géométriques**.

Afin d'obtenir les points de plus forte courbure, nous calculons une itération complète du squelette MB1-Hybride puis nous calculons les points extrêmes du squelette, ceux qui n'ont qu'un seul voisin dans l'image. De la même manière, les points de plus forte courbure négative sont calculés à l'aide de l'exosquelette[13]. La figure 4.9 montre un exemple de ce processus. Une seule itération de ce squelette suffit à produire un départ de branches suffisant pour notre étude.

La procédure que nous venons de proposer permet de calculer les points coins[14] convexes et concaves. Nous pouvons aussi calculer les points de jonction[15] en cherchant les points multiples du squelette. Cependant, par dualité, les points de jonction sont associés aux points coins du complémentaire, et les points multiples du squelette (resp. exosquelette) seront détectés par la présence de points extrêmes de l'exosquelette (resp. squelette), tant que des différents niveaux de connexité sont utilisés pour l'image et pour son

[13] le squelette du complémentaire de l'image
[14] L-points
[15] T-points

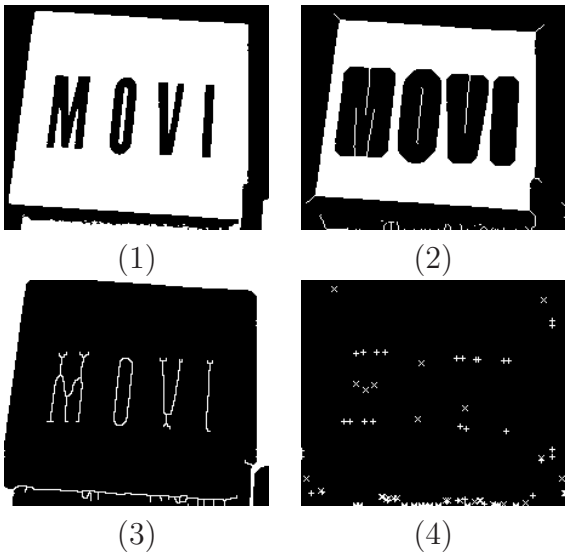



FIG. 4.9 – (1) L'image binaire original (2) le squelette 8-connecté (5 itérations) (3) l'exosquelette 4-connecté (5 itérations) et (4) l'union des points extrêmes du squelette (points marqués "x") et de l'exosquelette (points marqués "+").

complémentaire[16]. Par exemple, dans la configuration de la Figure 4.10, la jonction est détectée à la fois par la présence de points extrêmes de l'exosquelette 4-connecté (points marqués "+") ou par la présence de points extrêmes du squelette 4-connecté (points marqués "x").

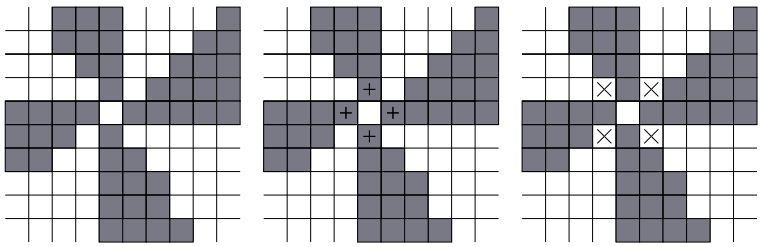


FIG. 4.10 – Dualité jonction/coin et le Théorème de Jordan en discret.

L'opérateur de calcul des points extrêmes utilisé dans cet algorithme est celui présenté Chapitre 2 Section 2.3.6. Bien entendu, nous utilisons sa version 4-connexe sur le squelette MB1 Hybride 4-connexe et idem en 8-connexité.

[16]Théorème de Jordan discret dans la maille carrée [KR89] et Définition 29

**Comparaison**

Afin de situer notre opŕateur par rapport aux détecteurs de points d'intérêt classiques nous avons mené une courte étude de comparaison en reprenant les images de tests et le protocole présenté par [RK06]. Ainsi nous effectuons le calcul de points dominants morphologiques sur les deux images "*Smith*" et "*Rosenthaler*" présentées dans leur articles respectifs ([SB97] et [RHKvH92]). Les algorithmes utilisés pour la comparaison sont les plus classiques, soit l'opérateur SUSAN-2D [AB86], un détecteur de Harris [HS88] et le détecteur de Förstner [FG87]. La Table 4.5 et la Figure 4.11 présentent les résultats obtenus lors de cette étude comparative.

| image Smith | | | | |
|---|---|---|---|---|
| | Asada | Harris | Forstner | Richefeu |
| nombre de points détectés | 61 | 49 | 38 | 61 |
| nombre de points corrects | 61 | 32 | 38 | 59 |
| nombre de fausses détections | 0 | 17 | 0 | 0 |
| image Rosenthaler | | | | |
| | Asada | Harris | Forstner | Richefeu |
| nombre de points détectés | 35 | 37 | 36 | 31 |
| nombre de points corrects | 30 | 31 | 31 | 28 |
| nombre de fausses détections | 5 | 5 | 0 | 3 |

TAB. 4.5 – Tableau récapitulatif de la comparaison des différents opérateurs de points d'intérêts.

Les résultats de ce comparatif nous permettent de dire que le détecteur morphologique obtient des performances comparables aux détecteurs classiques. On peut toutefois remarquer que si le nombre de points détectés est inférieur aux autres dans le cas de l'image de test de Rosenthaler, notre détecteur produit aussi moins de fausses détections. Cette particularité a été observée sur la plupart des images utilisés dans nos études et qui provient de l'utilisation du squelette morphologique produisant peu de branches induites par le bruit.

Pour être complet, ce comparatif aurait dû aussi donner une idée, au moins une estimation, du temps de calcul de chacun de ces algorithmes sur la rétine. Nous n'avons cependant pas eu l'occasion de porter d'autres techniques que celles reposant sur des outils morphologiques durant nos travaux pour effectuer cette évaluation. De même, nous n'avons pas eu le temps d'effectuer une étude plus approfondie, en particulier quantitative, sur les performances du détecteur en rotation et en changement d'échelle. Cependant, les résultats obtenus nous paraissent satisfaisants dans le cadre de notre travail de recherche.

**Conclusion**

La Figure 4.12 présente un exemple de résultat du calcul des points dominants morphologiques sur le banc algorithmique de la rétine. En terme de temps de calcul nous avons évalué le nombre d'opérations utiles pour effectuer le calcul des points dominants morphologiques pour des images codées sur 8 bits (255 niveaux de gris). Ainsi, l'algorithme complet nécessite 73 440 opérations ce qui correspond à un temps de calcul de 43 ms.

Bien que ce ne soit pas le sujet de cette thèse, nous nous devons de préciser que l'emploi de cet algorithme sur une architecture classique s'avère très complexe et nécessite un temps de calcul assez long. Il nous faut en effet effectuer autant de seuillages et de calcul de squelettes et d'exosquelettes que de niveaux de gris de l'image. Or, contrairement à une implantation rétinienne, le calcul séquentiel d'une itération du squelette est déjà coûteuse en nombre d'opérations. Nous avons cependant étudié la possibilité d'obtenir directement la fonction d'intérêt en utilisant les squelettes en niveaux de gris. Nous avons souligné alors que les squelettes en niveaux de gris n'offrent pas la même résistance au bruits que les squelettes MB binaires et ne permettent donc pas de localiser précisément les points de forte courbure.

La meilleure alternative séquentielle à notre détecteur reste certainement les détecteurs basés sur le produit de la courbure euclidienne par le gradient comme c'est la cas pour les opérateurs différentiels. Nous retrouvons alors des calculs semblables à ceux réalisés par des opérateurs classiques comme le détecteur de Harris.

### 4.4.3 Espace d'échelle

La théorie des espaces d'échelle est un cadre pour la représentation multi-échelle du signal. Le principe de causalité des espaces d'échelles est le fait qu'il n'y ait pas d'apparition de nouvelles structures alors que l'échelle augmente [HLS02] [BJ98]. Il a été montré par [CY89] que certaines opérations morphologiques, dont l'ouverture morphologique, ont la propriété de ne pas introduire de passage par zéro additionnels en évoluant dans l'espace d'échelle.

Une sélection hiérarchique des ensembles de niveaux définit un espace d'échelle morphologique [MM99]. Dans notre algorithme de calcul des points dominants, notre fonction d'intérêt est construite à partir de l'intégration de calculs effectués sur chaque ensemble de niveau. Le détecteur est donc compatible avec les espaces d'échelle morphologiques.

En effet, aucun nouveau point dominant n'apparaît lorsque l'échelle augmente.

Il est alors envisageable de fortement dégrader la dynamique de l'image sans pour autant altérer les résultats du calcul des points d'intérêt à l'aide d'outils morphologiques. Ainsi, une solution pour rendre le calcul des points dominants plus rapide est de décimer le nombre de niveaux de gris. La stratégie du choix des niveaux de gris à conserver joue un rôle important sur le résultat obtenu. De même que l'acquisition de l'image sur la rétine se fait selon une échelle logarithmique, il est judicieux de décimer les ensembles de niveaux selon une loi logarithmique.

La Figure 4.13 illustre les propriétés évoquées du calcul des points dominants morphologiques à travers un espace d'échelle construit par utilisation des filtres alternés séquentiels par reconstruction sur l'image d'origine.

Afin d'exploiter encore plus finement cette notion d'espace d'échelle, un mécanisme de mutualisation des pixels voisins appartenant à une primitive commune (région, contour, ligne de niveau) serait la bienvenue afin de permettre des calculs de plus haut-niveau. Nous pourrions alors mettre en commun les ressources mémoires de plusieurs pixels afin de stocker plus d'informations ou d'effectuer des algorithmes plus complexes. Un tel mécanisme, cependant, nécessite de disposer d'un mode de calcul asynchrone. Une étude d'une telle architecture a été menée conjointement à nos recherches par Valentin Gies[Gie05].

## 4.5 Conclusion et discussion

Nous avons vu dans ce chapitre le calcul de structures saillantes spatialement. Nous avons calculé ces points à l'aide d'un outil de squelettisation afin de tirer parti des avantages offerts par celui-ci.

Lors de nos travaux sur la détection du mouvement, nous avons introduit un nouvel opérateur, le filtre morphologique oublieux temporel (Chapitre 3 Section 3.5). Nous avons montré qu'il avait la particularité de détecter des pixels de fortes amplitudes de variation, des pixels saillants temporellement.

Nous avons utilisé une version particulière de cet opérateur hybride lors de la reconstruction hybride introduite dans le filtrage spatio-temporelle de la détection à base d'opérateur $\Sigma$-$\Delta$ (Chapitre 3 Section 3.6 Sous-Section 3.7.3). Un champ d'étude intéressant restant à explorer est le comportement de ces opérateurs en exploitant leurs propriétés à la fois temporelles et spatiales. Ainsi, nous pourrions envisager l'élaboration d'un détecteur

de mouvement basé sur un filtre de morphologie oublieuse spatio-temporelle, qui réagit à des structure saillantes spatio-temporelles.

Ces structures saillantes spatio-temporelles sont des points de focalisation visuelles dans une scène dynamique comme le sont les points d'intérêt. L'avantage d'un tel opérateur, outre de cumuler les fonctions de détection de mouvement et de calcul de zones d'intérêt directement en niveaux de gris, nous permettraient d'aborder la phase suivante du processus d'analyse du mouvement : l'estimation.

Dans le chapitre suivant nous présentons diverses techniques d'estimation du mouvement avant de proposer une méthode que nous avons choisie afin de suivre le déplacement d'un objet, par le biais de ses points dominants, dans une scène sur la rétine artificielle.

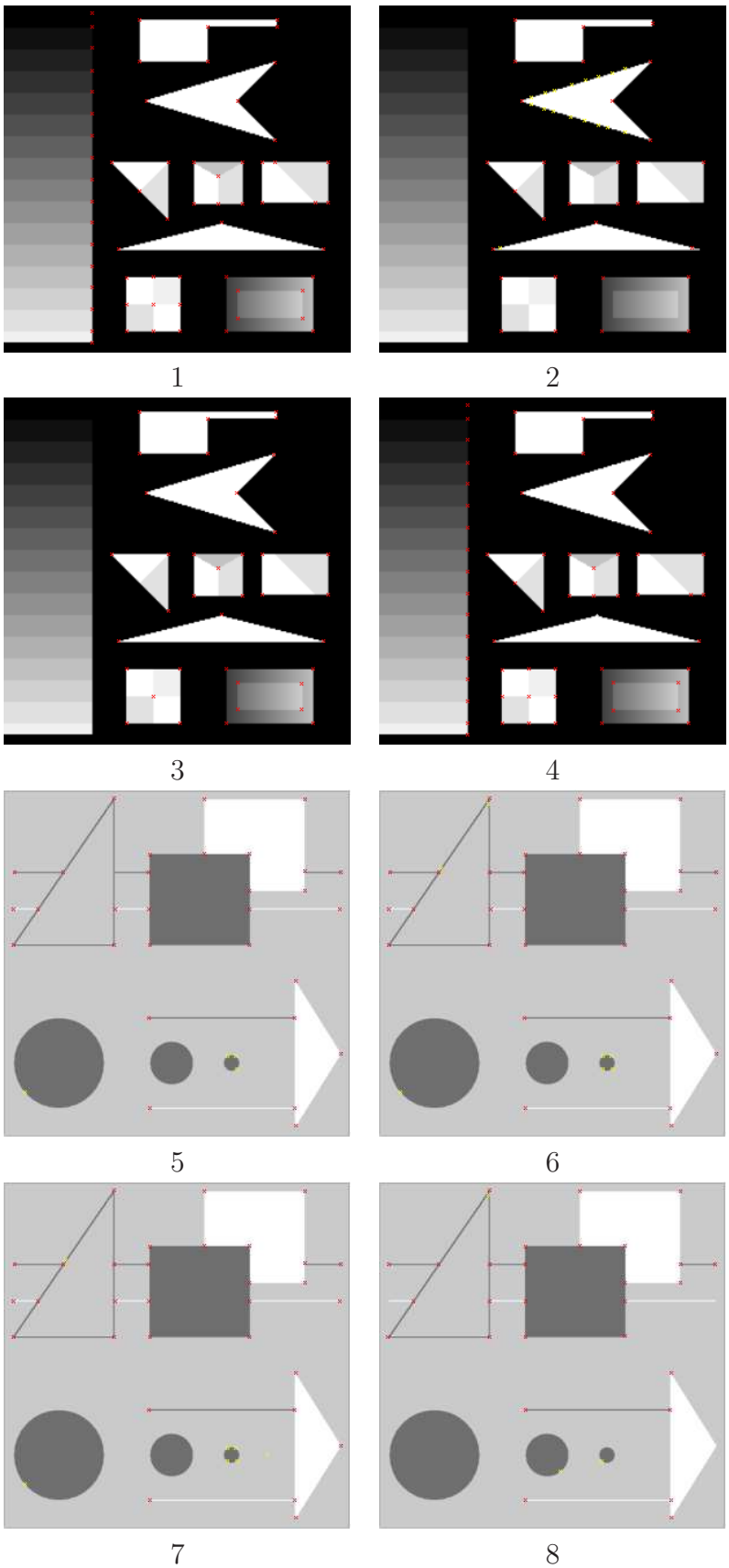



FIG. 4.11 – Résultats de la comparaison des différents opérateur de points d'intérets. L'image de Smith (1-4) et de Rosenthaler (5-8) avec les opérateurs suivants : 1 et 5) Asada et Brady, 2 et 6) Harris, 3 et 7) Förstner et 4 et 8) Points dominants morphologiques de Richefeu

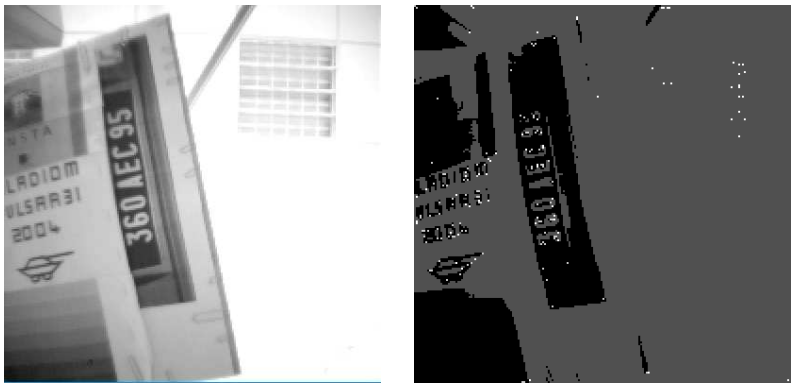


FIG. 4.12 – Exemple du calcul des points dominants morphologiques sur le banc algorithmique de la rétine.

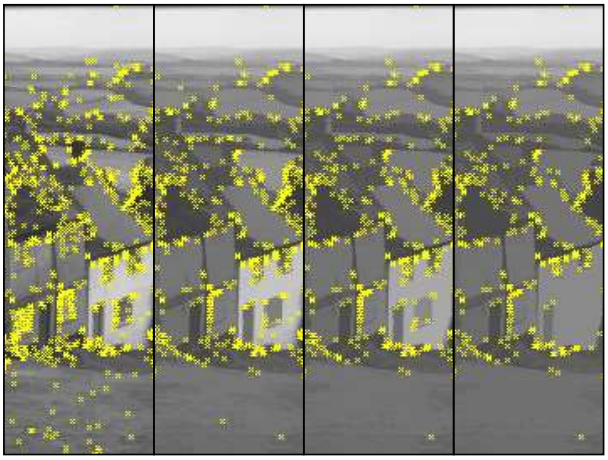


FIG. 4.13 – Le calcul de la détection des points dominants à travers un espace d'échelle morphologique sur un détail de l'image de test "goldhill".

# Chapitre 5

# Estimation du mouvement

## 5.1 Introduction

Dans ce chapitre, nous présentons les méthodes que nous avons utilisées afin d'estimer le mouvement. Ces techniques utilisent les primitives issues du calcul de structures saillantes afin de les mettre en correspondance entre deux trames consécutives. Ceci nous permet d'obtenir une estimation de la direction et de la vitesse des objets en mouvement dans la scène afin d'analyser ces résultats par un système tiers (hôte cortex). C'est-à-dire que, idéalement, les résultats de l'estimation du mouvement sont les seules données sortantes du circuit rétinien. Dans cette optique, nous nous sommes intéressés aux primitives issues du calcul des points dominants morphologiques.

L'analyse de séquences d'images peut-être considérée comme une suite d'analyses individuelles à effectuer assez vite pour tenir la cadence d'entrée des données, quitte ensuite à exploiter la cohérence inter-images pour affiner l'interprétation dans une deuxième phase. Une autre façon de voir les choses est de considérer le mouvement (et donc les différences d'images) comme la source primordiale d'information à exploiter pour l'analyse.

De cette distinction naît une première classification des algorithmes d'analyse du mouvement par le critère : "traiter avant ou traiter après ?"[Bon91][ZBM88].

- **"traiter avant"** : les propriétés statiques issues des méthodes de segmentation mono-image permettent de déduire les propriétés dynamiques par **mise en correspondance** de primitives entre images ;
- **"traiter après"** : les **propriétés dynamiques estimées** sur une séquence d'images permettent de retrouver des propriétés statiques, c'est-à-dire de déduire dans chaque

image l'existence, la position ou la forme des objets.
Notons qu'il s'agit de cas extrêmes, et que pour la plupart des algorithmes, distinction entre extraction de primitives et estimation est plus nuancée.

Nous présentons dans un premier temps un bref tour d'horizon des méthodes d'estimation du mouvement en se plaçant dans le contexte introduit auparavant. Nous évoquons notamment les différentes solutions de calcul de flot optique.

Puis nous exposons les techniques que nous avons employées afin d'estimer le mouvement dans une séquence d'images. Nous revenons plus particulièrement sur la mise en correspondance de structures saillantes, ici des points dominants morphologiques, au cours du temps. Se pose alors la question de décider à quel moment et de quels types seront les informations qui sortiront du circuit rétinien pour être analysées par un système externe.

Une des difficultés la plus connue en analyse du mouvement est le problème de l'ouverture[1]. La Figure 5.1 illustre schématiquement un cas où on est capable d'estimer localement le mouvement de plusieurs points mais incapable de décider à un niveau plus global du mouvement de l'objet.

Deux facteurs contribuent à l'ambiguïté du mouvement[Hil84] :
- la perte d'information due à la **projection d'un espace 3D sur une image 2D** : plusieurs surfaces 3D animées d'un mouvement différent peuvent avoir des champs projetés similaires sur le plan image.
- la perte éventuelle d'information due à la **distribution des intensités** : une sphère uniforme qui tourne sur elle-même n'engendre aucun changement de la fonction de luminance.

Nous pouvons distinguer deux approches visant à estimer le mouvement apparent :
- les **techniques locales** : le vecteur vitesse est estimé en chaque point en fonction de l'information spatiale et temporelle en ce point et dans son voisinage ;
- les **techniques globales** : le champ des vitesses est décrit par l'intermédiaire d'un modèle 2D paramétré permettant de connaître le vecteur de vitesse en tout point de l'image.

Dans une optique de suivi d'objets au cours du temps (poursuite de cibles), les techniques de corrélation normalisée surmontent les problèmes de variations d'intensité inter-images. Cependant aucune d'elles ne s'accommode des déformations par changement d'échelle, liées au rapprochement ou à l'éloignement d'une cible mobile rigide au cours du temps, et aux masquages partiels de la cible par un objet du relief, ce dernier cas produisant généralement deux pics de corrélation.

[1] *aperture problem* en anglais

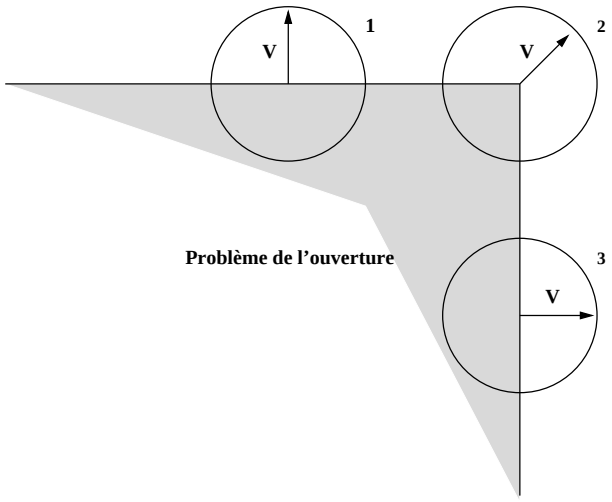



FIG. 5.1 – Problème de l'ouverture. Localement, en (1), (2) et (3) les mouvements $V$ (champs de déplacement) sont estimés dans des directions différentes. Nous ne pouvons cependant pas conclure sur le mouvement global de l'objet.

Un système de plots sous forme de cordonnées de boîtes englobantes a déjà été mis en place sur la rétine. Il s'agit d'extraire du circuit les coordonnées les plus précises possibles d'une cible afin d'analyser et d'identifier à l'aide d'un système classique disposant de capacité de calcul bien plus élevées. Cette expérience nous a permis de juger de la faisabilité de l'estimation sur la rétine et des techniques à employer pour y parvenir.

Idéalement, nous cherchons à estimer le mouvement des objets contenus dans une scène en nous basant sur le calcul de points dominants morphologiques (voir Figure 5.2). Cet algorithme consiste tout d'abord à effectuer un calcul de détection du mouvement afin de déterminer les régions à suivre, puis d'extraire de ces régions les points d'intérêt. Nous effectuons ensuite une étape de corrélation nous permettant d'estimer le sens et l'intensité du déplacement. Un certains nombre de rebouclage sont possible afin de rendre cet algorithme plus robuste. Il ne nous ait cependant aujourd'hui pas possible d'effectuer l'algorithme complet sur la rétine numérique en tenant la cadence vidéo et avec la quantité de mémoire dont nous disposons. Nous nous focalisons donc dans ce chapitre sur le dernière étape de notre algorithme, l'appariement des points d'intérêt que nous effectuerons sur l'ensemble des images originales. Nous reviendrons en fin de manuscrit sur les techniques à employer afin de pouvoir effectuer l'algorithme complet sur le circuit rétinien.

Le suivi de points d'intérêt dans une séquence d'images constitue un problème essentiel dans de nombreuses applications en vision par ordinateur [Mor79]. Un grand nombre

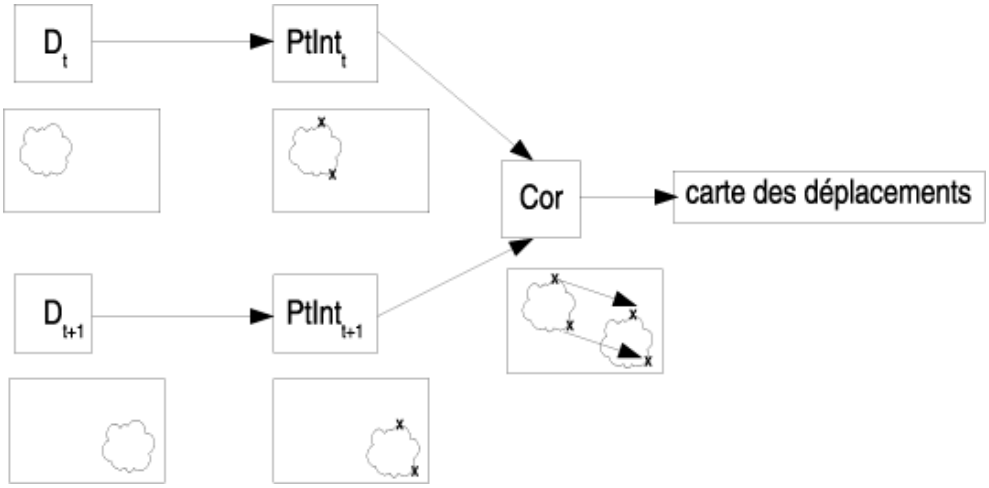



FIG. 5.2 – Algorithme complet de l'estimation du mouvement basé sur le suivi de point dominant

de tâches de haut niveau peuvent en dépendre directement comme par exemple la reconstruction 3D. Ce problème, qui consiste à reconstruire la trajectoire d'un point le long de la séquence, est un problème fondamentalement difficile. En effet, contrairement au suivi de formes structurées, les seules informations sur lesquelles nous pouvons nous appuyer sont des caractéristiques locales. Or, suivre un point au cours d'une séquence implique la construction d'une caractéristique locale invariante très lentement au cours de la trajectoire. Nous pouvons également noter qu'il est très difficile de construire au préalable un modèle dynamique du point, sans avoir une connaissance *a priori* du mouvement de l'objet environnant.

## 5.2 État de l'art

Les difficultés liées au problème de la poursuite de points ont amené les chercheurs à mettre en œuvre des techniques locales, basées sur des invariants géométriques et photométriques. L'hypothèse la plus couramment utilisée est la conservation de la luminance du point et de son voisinage le long de la trajectoire. La caractéristique locale considérée invariante est alors une petite fenêtre autour du point d'intérêt. Cette hypothèse a conduit à la définition de trois familles de méthodes.

La première comporte les méthodes intuitives basées sur des critères de similarité.

Ces méthodes, appelées **méthodes de corrélation**[2], sont utilisées dans de nombreux domaines pour le suivi de points mais aussi pour estimer les déplacements de formes hautement déformables comme les nuages en imagerie météorologique ou les écoulements fluides. Malgré le fait que les critères de similarité utilisés ne soient pas invariants aux transformations géométriques (changement d'échelle, rotation, distorsion perspective) et photométriques de l'image, ces méthodes restent très employées pour leur simplicité. Nécessitant une recherche exhaustive sur une région d'intérêt, elles peuvent être coûteuses en temps de calcul. Néanmoins, elles permettent de capter d'éventuels grands déplacements du point suivi.

Les **méthodes différentielles** représentent le second type d'approches. Elles sont élaborés à partir d'une formulation différentielle d'un critère de corrélation. L'exemple le plus représentatif et le plus connu est l'équation du **flot optique**, dérivée à partir de l'hypothèse de conservation de luminance dans un voisinage du point. Les méthodes différentielles sont en général rapides en terme de temps de calcul et basées sur la recherche de minimisation de coût, les rendant par conséquent sensibles aux minima locaux. Elles sont inadaptées lorsqu'il s'agit de traiter de grands déplacements. Il est toutefois possible de pallier ce dernier problème en utilisant une approche multi-résolution. Dans ce cas, nous ne considérons pas seulement la séquence d'images à sa résolution d'acquisition, mais nous construisons à partir de chaque trame une pyramide d'images successivement filtrées et sous-échantillonnées.

Enfin, la dernière catégorie regroupe les **méthodes fréquentielles** qui utilisent une famille de filtres spatio-temporelles accordées sur certaines fréquences (**filtres de Gabor**) pour extraire de l'information sur le mouvement. Le problème majeur de ces techniques est qu'elles sont coûteuses en terme de temps de calcul à effectuer afin d'obtenir de bons résultats.

### 5.2.1 Les méthodes de corrélation

La corrélation est une méthode bien connue en traitement du signal, qui est indifféremment applicable sur des signaux mono ou bidimensionnels. Elle nécessite la présence d'une forme caractéristique que l'on recherche de manière exhaustive dans deux images en faisant l'hypothèse d'un déplacement en translation. Elle s'exprime d'une manière canonique en deux dimensions par la formule :

$$C(u,v) = \int_{(x,y)\in\omega} f(x,y) f(x+u, y+v) dxdy$$

[2]aussi appelés méthodes de mise en correspondance, ou "block-matching" en anglais

Dans la pratique nous n'utilisons pas toujours l'opérateur de multiplication comme opérateur de comparaison, mais nous exprimons ce que nous appelons une fonction locale de ressemblance décalée des deux images centrées en $(x, y)$ par :

$$C_{x,y}(u, v) = \int_{(s,t)\in\Omega} F \circ f(x + s, y + t) * F \circ g(x + s + u.y + s + t)dsdt$$

où $\Omega$ est un voisinage de l'origine, $f$ et $g$ sont les deux images à comparer, le vecteur $(u, v)$ est le déplacement supposé, $*$ est l'opérateur de comparaison, $F$ est un opérateur que l'on peut appliquer aux images avant la comparaison (lissage ou autre régularisation) et $\circ$ est la composition des fonctions.

De nombreuses formulations sont possibles en fonction du signal ($F$) que l'on corrèle (image brute, image binaire ou Laplacien par exemple) et de la méthode d'appariement ($\circ$) utilisée (multiplication, valeur absolue des différences, fonction distance, etc. . . ). On retrouve ainsi, par exemple :

- Corrélation **directe** de l'image brute, sensible aux changements d'intensité en utilisant les opérateurs de multiplication ou les différences ;
- Corrélation **binaire** utilisant les images binaires obtenues par seuillage à partir de $f$ et $g$ ;
- Corrélation sur un **filtre Laplacien** éliminant les composantes d'ordre 0 et 1 de la différence d'image (valeur moyenne et pente) ;
- Corrélation **normalisée** par rapport aux moyennes ;
- Corrélation utilisant les **variances** normalisées ;
- Corrélation en calculant l'**erreur quadratique moyenne** ou la moyenne de la norme de l'erreur (recherche de minimum).

Les méthodes de corrélation dans l'optique de suivi de points d'intérêt caractérisent donc le point à suivre par une imagette qui contient la luminance du point et de son voisinage.

### 5.2.2 Les méthodes différentielles

les méthodes différentielles [BB95] sont fondées sur ce que l'on appelle l'équation de contrainte du mouvement apparent (ECMA), ou équation du flot optique :

**Définition 35 (Equation de Conservation du Mouvement Apparent)** *Soit une image I(x,y,t), on a par développement limité :*

$$\begin{aligned} I(x,y,t) &= I(x+V_x\triangle t, y+V_y\triangle t, t+\triangle t) \\ &= I(x,y,t) + \underbrace{\triangle t(V_x\frac{\delta I}{\delta x} + V_y\frac{\delta I}{\delta y} + \frac{\delta I}{\delta t})}_{=0} \end{aligned}$$

*d'où* $I_t + V.\nabla I = 0$

*avec* $V = (V_x, V_y)$*, le vecteur vitesse et* $I_t$ *la dérivée partielle du signal i par rapport au temps.*

Cette équation repose sur deux hypothèses :

1. la **conservation de l'intensité** au cours du mouvement ;
2. un **déplacement faible** entre deux images consécutives de la séquence.

Il est souhaitable que les déplacements restent faibles pour que la différentiation ait un sens. Cependant, une approche multi-résolution permet de considérer des déplacements d'ordre supérieur. L'équation seule ne peut-être résolue sans l'adjonction de contraintes supplémentaires pour garantir l'unicité de la solution. C'est le cas de la contrainte de régularisation employée par Horn et Schunck [HS80], dont la plupart des méthodes différentielles actuelles découlent [MB99].

Par définition, le flot optique est le champ de vitesse (ou de déplacement) produit dans le plan image par des objets en mouvement dans un espace 3D.

Nous présentons brièvement les principales méthodes de régularisation du flot optique selon Horn et Schunck puis Lucas et Kanade. Ce sont en effet les deux techniques le plus largement employées dans la communauté scientifique dans le cadre de l'étude du mouvement.

### Méthode de Horn & Schunck

Nous présentons dans cette section la méthode de Horn et Schunck afin de produire l'algorithme présenté Table 5.2. Nous nous sommes inspiré de cet démarche afin de produire notre propre algorithme de corrélation.

Soit une image $I(x,y,t)$, on pose les hypothèses classiques de l'ECMA :

$$\begin{aligned} I(x,y,t) &= I(x+\delta x, y+\delta y, t+\delta t) \\ &= I(x,y,t) + (\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t})\delta t \end{aligned}$$

Avec les composantes $\frac{\partial I}{\partial x} = I_x$, $\frac{\partial I}{\partial y} = I_y$ et $\frac{\partial I}{\partial x} = I_t$,on obtient :

$$I_x \delta x + I_y \delta y + I_t \delta t = 0$$

Ce qui nous amène à :

$$\begin{aligned} I_t + V.\nabla I &= 0 \\ V_x I_x + V_y I_y &= -I_t \\ (I_x, I_y).(V_x, V_y) &= -I_t \end{aligned}$$

Finalement, la contrainte d'intensité de l'image s'exprime par $F_H$ ($\Omega$ représente l'ensemble des positions spatiales) :

$$F_H = \int \int_{\Omega} (V_x I_x + V_y I_y + I_t)^2 dxdy$$

pour résoudre l'équation 35 du Chapitre 3, Horn et Schunck [HS80] introduisent une contrainte supplémentaire appelée contrainte de lissage de l'image. Mathématiquement, cette contrainte peut s'exprimer comme la minimisation de la quantité $F_S$ suivante :

$$F_S = \int \int_{\Omega} ((\frac{\delta u}{\delta x})^2 + (\frac{\delta u}{\delta y})^2 + (\frac{\delta v}{\delta x})^2 + (\frac{\delta v}{\delta y}))^2 dxdy$$

Horn et Schunck introduisent une fonction de coût dont le minimum est obtenu pour le champ de vitesse recherché. Cette fonction $F_C = F_H + \alpha^2 F_S$ est une combinaison des contraintes d'intensité et de lissage. $\alpha^2$ contrôle l'influence du terme de lissage. En dérivant $F_C$ selon $x$ puis selon $y$, on obtient le système suivant :

$$\left\{ \begin{array}{lcll} I_x^2 V_x + I_x I_y V_y &=& \alpha^2 \nabla^2 V_x - I_x I_t & (1) \\ I_y^2 V_y + I_x I_y V_x &=& \alpha^2 \nabla^2 V_y - I_y I_t & (2) \end{array} \right.$$

En effectuant $I_y(1) - I_x(2)$, on obtient :

$$\begin{aligned} & I_y(I_x^2 V_x + I_x I_y V_y) - I_x(I_y^2 V_y + I_x I_y V_x) = I_y(\alpha^2 \nabla^2 V_x - I_x I_t) - I_x(\alpha^2 \nabla^2 V_y - I_y I_t) \\ \Leftrightarrow \quad & I_y I_x^2 V_x + I_x I_y^2 V_y - I_x I_y^2 V_y - I_x^2 I_y V_x = I_y \alpha^2 \nabla^2 V_x - I_y I_x I_t - I_x \alpha^2 \nabla^2 V_y + I_y I_x I_t \\ \Leftrightarrow \quad & I_y \alpha^2 \nabla^2 V_x = I_x \alpha^2 \nabla^2 V_y \\ \Leftrightarrow \quad & \nabla^2 V_x \frac{I_y}{I_x} = \nabla^2 V_y \\ \Leftrightarrow \quad & \nabla^2 V_x = \frac{I_x}{I_y} \nabla^2 V_y \end{aligned} \tag{5.1}$$

Selon l'approximation du Laplacien $\nabla^2 V_x \simeq \bar{V_x} - V_x$[3], on a :

$$\Rightarrow \quad \nabla^2 V_x = \frac{I_x}{I_y}(\bar{V_y} - V_y)$$
$$\Rightarrow \quad \bar{V_y} = \frac{I_y}{I_x}\nabla^2 V_x + V_y$$

en remplaçant $\nabla^2 V_x$ dans (1), on obtient :

$$\begin{aligned}
I_x^2 V_x + I_x I_y V_y &= \alpha^2 \nabla^2 V_x - I_x I_t \\
\Leftrightarrow I_x^2 V_x + I_y^2 \nabla^2 V_x + I_x I_y V_y &= \alpha^2 \nabla^2 V_x + I_y^2 \nabla^2 V_x - I_x I_t \\
\Leftrightarrow I_x^2 V_x + I_x I_y (\frac{I_y}{I_x}\nabla^2 V_x + V_y) &= (\alpha^2 + I_y^2)\nabla^2 V_x - I_x I_t \\
\Rightarrow I_x^2 V_x + I_x I_y \bar{V_y} &= (\alpha^2 + I_y^2)(\bar{V_x} - V_x) - I_x I_t \\
\Leftrightarrow \alpha^2 V_x - \alpha^2 \bar{V_x} + I_x^2 V_x - I_x^2 \bar{V_x} + I_y^2 V_x - I_y^2 \bar{V_x} &= -I_x^2 \bar{V_x} - I_x I_y \bar{V_y} - I_x I_t \\
\Leftrightarrow (\alpha^2 + I_x^2 + I_y^2)(V_x - \bar{V_x}) &= -I_x(I_x \bar{V_x} + I_y \bar{V_y} + I_t) \\
\Leftrightarrow V_x - \bar{V_x} &= -I_x \frac{I_x \bar{V_x} + I_y \bar{V_y} + I_t}{\alpha^2 + I_x^2 + I_y^2} \\
\Leftrightarrow V_x &= \bar{V_x} - I_x \frac{N}{D}
\end{aligned}$$

avec $\begin{cases} N &= I_x \bar{V_x} + I_y \bar{V_y} + I_t \\ D &= \alpha^2 + I_x^2 + I_y^2 \end{cases}$

De même $V_y = \bar{V_y} - I_y \frac{N}{D}$, on obtient finalement la suite de la Table 5.1.

| **initialisation :** |
|---|
| $V_x^0 = 0$ |
| $V_y^0 = 0$ |
| **Répéter jusqu'à convergence :** |
| $\begin{cases} V_x^{k+1} &= \bar{V_x^k} - I_x \frac{N}{D} \\ V_y^{k+1} &= \bar{V_y^k} - I_y \frac{N}{D} \end{cases}$ |

TAB. 5.1 – Algorithme permettant le calcul du flot optique de Horn et Schunck.

Finalement, dans la pratique [BW05] [Sud01], on utilise l'algorithme de la Table 5.2, en prenant :

[3] $\bar{V_x}$ représente la valeur moyenne de $V_x$ sur le voisinage

$I_x(t) = \frac{1}{4}[I_E(t) + I_E(t+1) + I_{NE}(t) + I_{NE}(t+1) - (I(t) + I(t+1) + I_N(t) + I_N(t+1))]$,
$I_y(t) = \frac{1}{4}[I_N(t) + I_N(t+1) + I_{NE}(t) + I_{NE}(t+1) - (I(t) + I(t+1) + I_E(t) + I_E(t+1))]$,
et $I_t(t) = \frac{1}{4}[I(t+1) + I_N(t+1) + I_E(t+1) + I_{NE}(t+1) - (I(t) + I_N(t) + I_E(t) + I_{NE}(t))]$.

**Début**
  **Pour** $j := 1$ à $N$ faire
    **Pour** $i := 1$ à $M$ faire
    **Début**
      Calculer $I_x(i,j,t)$;
      Calculer $I_y(i,j,t)$ ;
      Calculer $I_t(i,j,t)$ ;
      Initialiser $V_x(i,j,t) = 0$ ;
      initialiser $V_y(i,j,t) = 0$ ;
    **Fin**
  Sélection du facteur de poids $\lambda$ ;
  Sélection de $n_0 \geq 1$ ;
  $n := 1$ ;
  **TantQue** $n \leq n_0$ faire
  **Début**
    **Pour** $j := 1$ à $N$ faire
      **Pour** $i := 1$ à $M$ faire
      **Début**
        $\bar{V_x} := \frac{1}{4}(V_x(i-1,j,t) + V_x(i+1,j,t) + V_x(i,j-1,t) + V_x(i,j+1,t))$ ;
        $\bar{V_y} := \frac{1}{4}(V_y(i-1,j,t) + V_y(i+1,j,t) + V_y(i,j-1,t) + V_y(i,j+1,t))$ ;
        $\alpha := \lambda \frac{I_x \bar{V_x} + I_y \bar{V_y} + I_t}{1 + \lambda(I_x^2 + I_y^2)}$ ;
        Calculer $V_x(i,j) := \bar{V_x} - \alpha . I_x(i,j,t)$ ;
        Calculer $V_y(i,j) := \bar{V_y} - \alpha . I_y(i,j,t)$ ;
      **Fin**
      $n := n + 1$ ;
  **Fin**
**Fin**

TAB. 5.2 – Algorithme du calcul du flot optique de Horn et Schunck.

### Méthode de Lucas et Kanade

On considére une série d'images $(I_k)_{k \in [0,n]}$. Entre deux images successives, sous l'hypothèse que la variation du niveau de gris est due uniquement au mouvement de la caméra

et/ou du point $s$ suivi, la conservation de la luminance s'écrit :

$$I_k(s) = I_{k+1}(\varphi(s))$$

où $\varphi(s)$ représente le pixel déplacé. Cette hypothèse est très forte, elle est rarement vérifiée pour tous les pixels de l'image. Afin de la relâcher, on recherche une fonction spatiale décrivant le déplacement qui minimise la valeur du résidu de $SSD$[4] sur un voisinage.

**Définition 36 (Somme des différences au carré)** *Soit un pixel $s$ de l'image $I$, la somme des différence au carré de $s$ dans la fenêtre $\mathcal{W}$ autour du point $s$ est :*

$$\epsilon = SSD(s) = \sum_{f \in \mathcal{W}} [I_{k+1}(\varphi(s)) - I_k(s)]^2$$

Dans le cas de faibles déplacements, l'hypothèse permettant d'approcher le mouvement par une simple translation est admise (donc $\varphi(s) = s+d$). Par un développement de Taylor à l'ordre 1, nous pouvons faire l'approximation suivante :

$$I_{k+1}(s+d) \simeq I_k(s) + g_k(s)^t.d + \partial_k I_k(s)$$

où $g_k(s)^t = [\partial_u I_k(s), \partial_v I_k(s)]$ est le gradient spatial de $I_k(s)$, et $\partial_k I_k$ est la dérivée par rapport au temps. Lorsque le vecteur de déplacement $d$ est faible entre chaque image, l'expression du résidu peut donc être réécrite sous la forme :

$$\epsilon = \sum_{\mathcal{W}} [g_k.d - h_k]^2$$

avec $h_k = I_{k+1}(s) - I_k(s)$. Afin de minimiser ce résidu, nous recherchons la translation $\hat{d}$ qui rend la dérivée de $d$ par rapport à $e$ nulle. Nous obtenons le système linéaire :

$$Gd = e$$

avec $G = \sum_{\mathcal{W}} g_k g_k^T d$ et $e = \sum_{\mathcal{W}} h_k g_k d$. Une solution à ce système linéaire est $\hat{d} = G^{-1}e$.

Cette méthode est largement utilisée et donne de bons résultats dans le domaine du suivi de cibles. Elle peut également être employée pour l'estimation de champs denses de mouvements. Tout comme les méthodes de corrélation, un des problèmes importants des méthodes différentielles est leur sensibilité aux changements d'illumination. En effet, un tel changement ne respecte pas la conservation de la luminance le long de la trajectoire qui était l'hypothèse de départ [LK81] [TK01].

[4]Somme des différences au carré, voir Définition 36

### 5.2.3 Les méthodes fréquentielles.

Ces méthodes sont apparues suite à la constatation que certaines propriétés du mouvement sont plus accessibles dans le domaine fréquentiel. En effet les filtres fréquentiels comme le **Filtre de Gabor** agissent comme des filtres passe bande orientés dans la direction du mouvement. Le principe de base est de trouver une équation équivalente à l'équation du mouvement, dans le domaine fréquentiel. Cela suppose que le champ de vitesses soit localement translationnel. Ainsi, après un calcul de transformée de Fourier, nous obtenons l'équation $\omega_x u + \omega_y v + \omega_t = 0$, où $\omega_x$, $\omega_y$ et $\omega_t$ sont les fréquences selon l'axe des $x$, l'axe des $y$ et l'axe des $t$. Cette équation de contrainte du mouvement dans le domaine fréquentiel se traduira par le fait que l'énergie spatiale sera contenue dans un plan passant par l'origine et dont l'orientation est liée à la translation 2D cherchée.

Pour déterminer un tel plan, des filtres sensibles à une direction de mouvement donnée sont utilisés. Ainsi, [Hee87] a utilisé des filtres de Gabor spatio-temporels pour estimer la vitesse dans l'image.

**Définition 37 (fonction de Gabor)** *Une fonction de Gabor $G$ se définit comme une fonction gaussienne modulée par une onde sinusoïdale :*

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2})} e^{-j2\pi(xf_{x0}+yf_{y0})}$$

*avec $\sigma_x$ et $\sigma_y$ la largeur respectivement selon $x$ et $y$ de la fonction gaussienne et $f_{x0}$ et $f_{y0}$ les fréquences spatiales de la modulation.*

Ce filtre, de type passe-bande orienté, a une réponse impulsionnelle complexe. La fréquence centrale $f_0$ est donnée par $\sqrt{f_{x0}^2 + f_{y0}^2}$, l'orientation $\theta$ par $arctan(\frac{f_{x0}}{f_{y0}})$ et la largeur de bande par $\sigma_x$ et $\sigma_y$. La réponse fréquentielle du filtre de Gabor est :

$$TF(G)(f_x, f_y) = e^{-2\pi^2(\sigma_x^2(f_x-f_{x0})+\sigma_y^2(f_y-f_{y0}))}$$

Les filtres de Gabor sont d'autant plus sélectifs en fréquence que leur support spatial est large, et réciproquement. Considérons un banc de $N$ filtres de Gabor où les filtres sont isotropes ($\sigma = \sigma_x = \sigma_y$), positionnés sur une couronne définie par la fréquence centrale $f_0$, chaque filtre ayant pour orientation $\theta_i = \frac{i\pi}{N}$. Ce banc de filtres est noté $G_{f_0,\sigma,N}$.

**Définition 38 (Filtre de Gabor)** *Le filtre de Gabor de la $i^{eme}$ sous-bande a pour réponse impulsionelle $G_i(x, y)$ :*

$$G_i(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} e^{-j2\pi f_0(xcos\theta_i + ysin\theta_i)}$$

Le réglage des paramètres $f_0$, $\sigma$ et $N$ est déterminant pour estimer de façon précise le mouvement. Ces trois paramètres influent en particulier sur le choix de calcul de filtres sur l'image (gradients), sur le conditionnement du système d'équations du flot optique, ainsi que sur le coût calculatoire de l'opération.

La décomposition d'images à canaux multiples est un ensemble de sous-bandes :

$$I_{G_i}(x,y,t) = I(x,y,t) * G_i(x,y) \ , \quad i = 1...N$$

où $N$ est le nombre de canaux et $*$ dénote la convolution en deux dimensions.

A une position $(x, y)$ donnée, chaque $I_{G_i}$ satisfait l'équation de la Définition 38. On aboutit au système surcontraint suivant :

$$\underbrace{\begin{pmatrix} \Omega_1^x & \Omega_1^y \\ \Omega_2^x & \Omega_2^y \\ & \vdots \\ \Omega_N^x & \Omega_N^y \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} u \\ v \end{pmatrix}}_{V} = - \underbrace{\begin{pmatrix} \Omega_1^t \\ \Omega_2^t \\ \vdots \\ \Omega_N^t \end{pmatrix}}_{B}$$

où :

$$\Omega_i^x = \frac{\delta I_{G_i}}{\delta x}, \ \Omega_i^y = \frac{\delta I_{G_i}}{\delta y}, \ \Omega_i^t = \frac{\delta I_{G_i}}{\delta t}$$

Dans ce cas, l'estimation du mouvement consiste à résoudre le système précédent pour $(x, y)$[BP00]. Dans le domaine fréquentiel, le filtre de Gabor se comporte comme un modèle de filtre passe-bande laissant certaines fréquences. Les filtres de basses fréquences ont une réponse élevée pour les motifs qui se déplacent rapidement et les filtres de hautes fréquences ont une réponse élevée pour les motifs qui se déplacent lentement.

Parmi les avantages de ces méthodes on pourrait énoncer : un lissage intrinsèque qui réduit les effets du problème d'ouverture, une meilleure robustesse au bruit et des résultats de qualité pour des séquences d'images naturelles. Par contre, un coût élevé en calcul et le manque de précision à proximité des frontières du mouvement en sont ses défauts.

### 5.2.4 Conclusion

Pour une revue générale des travaux qui ont été réalisés dans le domaine de l'estimation du mouvement et la mise en correspondance, le lecteur pourra se reporter aux travaux suivants [GCBV93] [BB95] [Hen96] [GCN+98] [Bru01] [Arn01] .

Les méthodes différentielles, basées sur la résolution de l'équation du flot optique, présentent de nombreux avantages face aux méthodes de mise en correspondance et face aux méthodes fréquentielles. L'équation du flot optique permet une estimation subpixellique directe du mouvement, contrairement aux méthodes de mise en correspondance. La mesure du mouvement ne nécessite qu'un calcul local des dérivées spatio-temporelles de la séquence. Ces opérations ont un coût de calcul faible comparées aux filtrages spatio-temporels imposés par les méthodes fréquentielles. Ces deux avantages, associés au fait que l'équation du flot optique est linéaire par rapport au vecteur vitesse, expliquent le succès et le nombre très important de travaux portant sur les méthodes différentielles.

En effet, le calcul du flot optique de type Horn et Schunck est certainement moins coûteux que la mise en correspondance directe sur une architecture courante. Cependant, elle nécessite l'emploi de primitives algorithmiques difficiles à mettre en place sur le système de vision rétinien comme le sont le calcul de différence employant des divisions, par exemple. Voilà pourquoi nous avons fait le choix de techniques employant des opérateurs utilisant des primitives de calculs simples pour la rétine au détriment du nombre d'opérations effectuées entre deux trames consécutives. Nous exposons leur principe dans la section suivante.

## 5.3 Mise en correspondance de structures saillantes

Dans cette section, nous présentons la méthodologie employée afin de mettre en correspondance des pixels particuliers entre deux images consécutives d'une séquence. Afin de déterminer les points à suivre, nous utilisons les points dominants morphologiques présentés dans le Chapitre précédent. Les pixels ainsi choisis correspondent à des régions de la scène où se trouvent des structures saillantes spatiales (comme des coins, des points multiples, etc. . . ).

Afin de répondre aux contraintes imposées par le système rétinien, nous employons une technique de corrélation directe entre images brutes. Toutefois, notre recherche est guidée par le résultat du calcul des points dominants. Ceci nous permet de réduire considérablement le nombre de points à suivre mais aussi à s'adapter aux conditions

de la scène ( éclairement, mouvement propre de l'objet, bruits, etc. . . ).

Le champ de déplacement $V(V_x, V_y)$ de points de $I_t(x, y)$ dans $I_{t+1}(x, y)$ est testé dans le **voisinage de corrélation** $D$ qui correspond à la fenêtre d'intégration dans laquelle est effectuée la mesure de vraisemblance. Cette fenêtre de corrélation est ensuite déplacée dans un voisinage autour du point nommé **voisinage de recherche** $\Omega$. La mesure de vraisemblance $C$ associée à $(V_x, V_y)$ est :

$$C(V_x, V_y) = \sum_{i \in D} \sum_{j \in D} |I_t(x + i, y + j) - I_{t+1}(x + V_x + i, y + V_y + j)|$$

Pour d'obtenir le déplacement le plus vraisemblable, il faut minimiser cette fonction $(argmin\, C(V_x, V_y))$.

**Pour** chaque pixel $P$ de $I_t$ **{**
  **Si** la fonction d'intérêt $f_{int}(I_{det}(P)) > S$ **{**
    $Vx = 0\,;\ Vy = 0\,;\ \Sigma_{min} = +\infty\,;$
    **Pour** chaque point $d = (d_x, d_y)$ du voisinage de recherche $\Omega$ **{**
      $\Sigma$=0 ;
      **Pour** chaque point $(u, v)$ de l'espace de corrélation $D$
        $\Sigma = \Sigma + |I_t(x + u, y + v) - I_{t+1}(x + u + d_x, y + v + d_y)|;$
      **Si** $(\Sigma < \Sigma_{min})$
        $\Sigma_{min} = \Sigma\,;\ V_x = dx\,;\ V_y = dy\,;$
    **}**
  **}**
**}**

Tab. 5.3 – Algorithme de mise en correspondance de points d'intérêt.

Le principe de la mise en correspondance est décrit dans l'algorithme de la Table 5.3. Typiquement, le voisinage de recherche doit être large afin de suivre des mouvements de large amplitude mais l'espace de corrélation, lui, doit être plus réduit afin de limiter le temps de calcul et aussi d'être moins sensible aux rotations.

Sur la rétine, la somme des différence entre les pixels est la plus coûteuse en terme de nombre d'opérations et donc de temps de calcul. Pour des images en 8 niveaux de gris, la seule différence entre un point et son voisin direct représente 8 déplacements élémentaires (différence bit à bit), c'est-à-dire au moins 16 lectures et 16 écritures. Ce chiffre double pour un voisin diagonal (correspondant à deux déplacements directs) et triple dans le pire des cas pour un point distant de seulement deux unités dans la maille carrée (le voisin direct d'un voisin direct).

Ainsi, afin de limiter le nombre d'opérations effectuées, il nous faut trouver une stratégie pour calculer la somme des voisins de l'espace de corrélation. Nous pouvons ici utiliser les caractéristiques architecturales de la rétine à notre avantage en tirant parti du mode de calcul massivement parallèle. Ainsi, nous pouvons effectuer la somme sur un voisinage $8 \times 8$ (non centré) en seulement 6 opérations (additions). La Table 5.4 et la Figure 5.3 présentent l'algorithme utilisé pour effectuer cette somme.

$$
\begin{array}{l}
SomVoisDir(D)\{ \\
D = D + D.E \\
D = D + (D.E).E \\
D = D + (((D.W).W).W).W \\
D = D + D.N \\
D = D + (D.N).N \\
D = D + (((D.S).S).S).S \\
\}
\end{array}
$$

TAB. 5.4 – Algorithme de calcul de la somme des voisins dans un carré $8 \times 8$ non centré. La Figure 5.3 présente un schéma de ce calcul

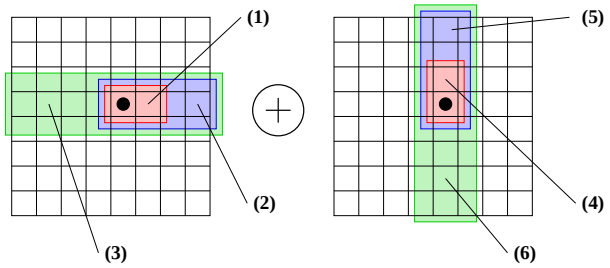



FIG. 5.3 – Schéma de l'algorithme de calcul de la somme des voisins dans un carré $8 \times 8$ non centré proposé dans la Table 5.4. Les numéro de la figure correspondent à l'ordre des opérations menées pour exécuter l'algorithme.

Afin de rendre le parcours de l'espace de recherche plus efficace, nous avons choisi de mettre en place un parcours en spirale, qui nous permet de n'effectuer qu'une seule translation élémentaire par itération (dans une des quatre directions 4-connexes). Nous ne passons donc qu'une seule fois en chaque point et économisons donc du temps de calcul en termes de déplacement dans l'image (translations à effectuer).

Le principe du parcours spirale est de commencer dans une direction et d'avancer d'un pas puis de prendre la direction suivante et d'avancer encore d'un pas. On prend ensuite la troisième direction et on avance de deux pas, puis on prend la quatrième et dernière

direction et on avance de deux pas. On reprend alors la première direction mais on avance de trois pas, et ainsi de suite .... On fait un pas de plus tous les deux changements de directions (N,E,SS,WW,NNN,EEE,...). La Figure 5.4 représente ce parcours.

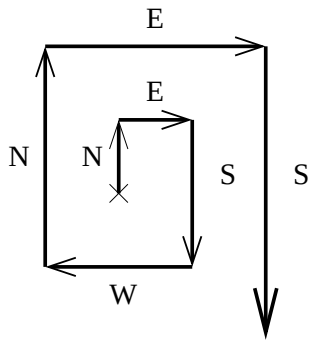



FIG. 5.4 – Représentation schématique du parcours spirale.

La Table 5.5 présente l'algorithme de corrélation tel qu'il a été implanté sur la rétine. La variable $\Delta_{min}$ nous permet de stocker la somme minimale des voisins correspondant au pixel de $I_{t-1}$ le mieux corrélé au pixel courant de $I_t$. Etant donné que les pixels sur la rétine sont codés sur 6 bits et que l'on utilise un **compteur saturant**[5] pour effectuer la somme des voisins, la plus grande valeur possible pour $K$, exprimée en binaire, est $[011111]_2$, on initialise donc $\Delta_{min}$ à $[100000]_2$.

Comme le montre la Table 5.6, en ajoutant les deux lignes de variables dédiées au calcul et celle liée aux variables intermédiaires ainsi que les registres de communications de la rétine, l'algorithme de corrélation occupe toute la mémoire disponible en codant seulement les pixels sur 6 bits. Nous avons mesuré expérimentalement le temps de calcul pour un espace de recherche $D = 20$ sur une rétine cadencée à 5 Mhz. Le temps mesuré pour une fenêtre $10 \times 10$ ( 199 130 instructions de pseudo-code), est de 400 ms entre deux trames dont 40 ms pour l'acquisition seule de l'image et 100 ms pour l'extraction des résultats, soit 260 ms pour le calcul. Pour comparaison, en effectuant uniquement le calcul des points d'intérêt, la rétine ne prend que 43 ms pour un peu moins de 70 000 opérations effectuées en codant les images sur 8 bits.

Actuellement, à la fin de la procédure de corrélation, une image correspondant aux corrélations minimales[6] pour chaque pixel de l'image est extraite de la rétine. Afin de

[5]La somme est codée sur un nombre de bits représentatifs. Les autres ne sont pas codant et ne servent qu'à indiquer que la somme a saturée. Cette technique est très utile pour borner un résultat et faire l'économie de quelques registres et de quelques instructions.

[6]représentées par le couple composé par un point d'intérêt dans l'image $I_t$ et le point obtenu de score de corrélation minimal dans $I_{t+1}$

Entrée : $I_t$ et $I_{t-1}$
$J_t = \text{PtInt}(I_t)$ ;
$J_t = \delta(J_t)$ (seuillage de la fonction d'intérêt) ;
$nb = 0$ ;
$\Delta_{min} = [10000]_2$ ;
**Pour** $(j = 0\,;\, j < D\,;\, j{+}{+})$ **{**
**Pour** $(i = 0\,;\, i < nb\,;\, i{+}{+})$ **{**
**Si** $((j \, mod \, 4) = 0)$ **alors** $dir = Est$
**sinon Si** $((j \, mod \, 4) = 1)$ **alors** $dir = Nord$
**sinon Si** $((j \, mod \, 4) = 2)$ **alors** $dir = Ouest$
**sinon** $dir = Sud$ ;
$I_{t-1}$=$I_{t-1}.dir$ ;
$\Delta = I_t - I_{t-1}$ ;
$\Delta = |\Delta|$ ;
$K = SomVoisDir(\Delta)$ ;
$K - \Delta_{min}$ ; (on sauvegarde le bit de signe $S$)
$\Delta_{min} = K \times (J_t \wedge S) + \Delta_{min} \times (\overline{J_t} \vee \overline{S})$ ;
**}**
**Si** $((j \, mod \, 2)! = 0)$ **alors** $nb{+}{+}$ ;
**}**
**}**

TAB. 5.5 – Algorithme de corrélation implanté sur la rétine numérique.

| module | fonction | INS | REG |
|---|---|---|---|
| Acquisition | "OU exclusif" | 262 | 9 |
| Point d'intérêt | inversion | 8 | 16 |
| | Squelette | 99+94 | 19 |
| 255 × | Point Extrême | 28+16 | 17 |
| | "OU" | 1 | 1 |
| | addition | 30 | 8 |
| Sous-Total | - | 69 870 | 19 |
| Corrélation | translation | 8 | 27 |
| | soustraction | 30 | 33 |
| | valeur absolue | 19 | 34 |
| $D*(D+1)\times$ | somme des voisins | 192 | 34 |
| | soustraction | 30 | 34 |
| | maj | 30 | 34 |
| Sous-Total | - | $288*(D*(D+1))$ | 34 |
| Total | - | $69870+(288*(D*(D+1)))$ | 34 |

TAB. 5.6 – Chiffrage de l'algorithme de corrélation sur la rétine en terme de nombre d'opérations et d'occupation de la mémoire pour des pixels codés sur 6 bits.

rendre cette phase optimale, il serait plus judicieux de ne sortir que des "*flags*"[7] pour les pixels correspondant aux points d'intérêt lorsque la somme est inférieure au $\Delta_{min}$ courant. Dès lors, c'est le cortex qui détecterait ces "*flags*" et en déduirait le déplacement correspondant.

## 5.4 Conclusion

Nous avons montré lors de cette étude qu'il est raisonnable d'élaborer un algorithme de mise en correspondance de points d'intérêt sur le circuit rétinien en temps réel. Afin de rendre cette méthode plus robuste et d'aller plus loin dans l'analyse des objets détectés et d'estimer le mouvement, il manque un mécanisme de dialogue rétine-cortex plus élaboré. Il permettrait dans un premier temps le rebouclage des mouvements estimés avec la phase de détection et, dans une seconde partie, il intégrerait une phase d'analyse des objets en mouvement.

Nous avons envisagé une variante de cet algorithme de corrélation en se basant sur les

[7] variables booléennes

contours morphologiques exposés au Chapitre 2 (Partie 2.3.5) au lieu d'utiliser les points d'intérêt. Un filtrage complémentaire permet de réduire les images issues de ce mécanisme en un point. Cette technique nous permet de remplacer la partie de code correspondant au calcul des points d'intérêt, assez coûteuse, par celle de calcul d'opérations morphologiques simples qui le sont beaucoup moins. Le gain en taille de code nous permet ainsi d'ajouter un algorithme de détection du mouvement plus complexe.

Le chapitre suivant traite des nombreuses optimisations que l'on peut apporter à un code rétinien et que nous avons abondamment utilisées, notamment lors de la conception de l'algorithme de corrélation.

# Chapitre 6

# Plateforme de développement algorithmique

## 6.1 Introduction

L'élaboration d'algorithmes pour le circuit rétinien est une tâche ardue, car originale dans le monde de la programmation sur ordinateur, sans connaissance technique préalable ni d'outil d'aide au développement. Nous présentons dans ce chapitre des méthodes mises en place au cours de la thèse et les logiciels conçus afin de développer des algorithmes rétiniens et d'en rendre l'écriture plus agréable.

Ce chapitre est scindé en deux parties correspondant au découpage typique du développement d'algorithmes pour la rétine numérique.

La première partie traite des outils de conception, de simulation et de validation du code rétinien. Il s'agit de tous les logiciels développés au cours de la thèse qui nous ont permis de passer des réflexions théoriques à propos de la détection du mouvement aux simulations réalisées sur un émulateur du circuit rétinien.

La seconde partie concerne les techniques d'optimisation et de gestion du code sur la rétine. Il s'agit d'un ensemble de méthodes de programmation permettant d'être plus efficace dans l'écriture d'algorithmes pour rétines numériques mais aussi d'un cadre de travail dont l'objectif est de permettre au programmeur de travailler à un niveau d'abstraction plus élevé ( généricité du code, allocation de la mémoire, ...).

## 6.2 Outils de conception, de simulation et de validation

### 6.2.1 Introduction

Nous avons élaboré un certain nombre de logiciels qui nous ont permis de valider et d'améliorer les algorithmes conçus pour la rétine. Nous présentons donc ici les outils de validation et de conception écrits sur une architecture classique.

L'objectif de cette section est de déterminer un cadre méthodologique de conception des algorithmes pour la rétine. D'un point de vue technique, nous avons choisi d'utiliser le langage de programmation $C++$[1] car celui-ci nous permet de manipuler des objets informatiques. Nous avons ainsi la possibilité d'écrire des algorithmes plus facilement et plus efficacement. De plus, ce langage a une grande robustesse et des performances éprouvées héritées du $C$[2]. Il nous offre la possibilité de créer des objets et des fonctions associées à ces objets avec de bonnes performances, nous permettant de simuler une architecture massivement parallèle en temps raisonnable sur des images de petite taille (de taille maximum 320x200 pour des algorithmes peu complexes, donc nécessitant un nombre d'instructions limité).

Nous avons aussi fait le choix d'utiliser une bibliothèque graphique spécifique, *Qt*, développée par la société Trolltech[3] pour élaborer des interfaces graphiques rapidement. Cette bibliothèque a l'avantage de permettre d'obtenir un code multiplateforme, efficace et directement interfaçable en $C++$. Nous utilisons des interfaces graphiques afin de visualiser le résultat des algorithmes et de tester leurs paramétrages.

Nous aurions pu utiliser d'autres langages, notamment *Tcl/Tk*[4] ou Python[5], qui

[1] Bjarne Stroustrup a développé le C++ au cours des années 1980, alors qu'il travaillait dans le laboratoire de recherche *Bell* d'*AT&T*. Il s'agissait en l'occurrence d'améliorer le langage C. Les premières améliorations se concrétisèrent par l'ajout du support des classes, suivies par de nombreuses autres comme les fonctions virtuelles, la surcharge d'opérateurs, l'héritage (simple ou multiple), les "*templates*", la gestion d'exceptions, ...

[2] Le langage C est apparu au cours de l'année 1972 dans les Laboratoires *Bell*. Il était développé en même temps que **UNIX** par Dennis Ritchie et Ken Thompson. Par la suite, Brian Kernighan aida à populariser le langage

[3] http ://www.trolltech.com

[4] *Tool Command Language*, généralement abrégé Tcl, est un langage de programmation initialement conçu en 1989 pour le système d'exploitation **UNIX** par John Ousterhout et son équipe à l'Université de *Berkeley*

[5] Créé en 1989 par Guido van Rossum, ce langage trouve l'origine de son nom dans la série télévisée

offrent des librairies de traitement d'image et de conception d'interfaces homme-machine éprouvées, mais nous avions une bonne connaissance préalable de *Qt* et nous souhaitons pouvoir distribuer certains de nos logiciels sans nous soucier des plateformes hôtes pour nos programmes.

Dans un premier temps nous présentons quatre logiciels développés à l'aide du couple $C++/Qt$. Le premier est un émulateur de rétine bas niveau (emulRet) qui reproduit le comportement d'une rétine numérique depuis l'acquisition d'images jusqu'à la sortie de données préformatées.

Le second outil, TraitLib, est un logiciel permettant de tester et de valider les algorithmes de traitements d'images statiques. Il nous a permis de comparer les différentes méthodes de filtrages linéaires, de calcul de points d'intérêt ou encore de squelettes morphologiques ainsi que ceux en niveaux de gris.

Le troisième outil développé, TraitMotion, nous aide à concevoir des algorithmes de détection et d'analyse du mouvement. Il travaille à partir de séquences de tests rejouées depuis le disque dur mais aussi à partir d'une caméra numérique ou d'une webcam interfacée à l'ordinateur.

Le dernier outil permet la visualisation du champ de déplacement calculé à l'aide de techniques de corrélation.

### 6.2.2 Etude préliminaire

Nous avons été très vite confrontés au cours de nos travaux à un problème de base pour tous les chercheurs et les ingénieurs en traitement d'image : le choix du format d'image. Nous avons besoin d'images le moins bruitées possible donc n'ayant pas subi la moindre compression avec perte. Nous souhaitons pouvoir disposer de la plus grande bibliothèque d'images sans avoir à convertir les images avant de les traiter.

La bibliothèque *Qt* offre un large éventail de format d'images et un accès facile au contenu brut de celle-ci. Du côté du **programmeur**[6] nous avons choisi de travailler avec des images au format bitmap[7] alors que pour l'**utilisateur**[8], tous les formats utilisés

humoristique des *Monty Python*

[6]La fonction d'un programmeur est la création d'algorithmes et la conversion d'algorithmes dans un langage de programmation

[7]BMP sous Windows et PBM sous les Unix

[8]L'utilisateur est celui qui utilise l'ordinateur mais qui n'est pas nécessairement informaticien

par $Qt$ sont valides. Seul le format $GIF$[9] nous a posé problème à cause du caractère propriétaire[10] de l'algorithme de compression $LZW$[11] utilisé dans ce format mais nous avons pour ce cas écrit notre propre outil de décompression[12].

Ainsi, quelque soit l'outil développé, le processus d'acquisition de l'image est toujours le même :

1. ouverture de l'image au format $X$ avec l'objet $Qt$ pixmap ;
2. enregistrement du contenu du pixmap dans l'image originale au format $BMP$ ;
3. ouverture de l'image $BMP$ ;
4. création d'un objet $Img$ à partir des données brutes de l'image.

Les points (2) et (3) sont facultatifs et nous servent à revenir en arrière et à réouvrir l'image originale rapidement et sans surcharger la mémoire.

La classe Img qui manipule les données de l'image est utilisée par toutes les fonctions de traitement d'images développées pour les outils décrits ci-après. Elle comporte cinq données membres renseignant sur le type d'opérations à mener en bordure de l'image, son éventuelle déplacement[13], la longueur de l'image, sa largeur et un pointeur vers ses données. Les données sont organisées dans une matrice créée dynamiquement de taille longueur × largeur de type *unsigned char*[14]. Selon le protocole du format d'image BMP, lorsque l'on écrit le tableau dans un fichier de type BMP, si la largeur de l'image n'est pas un multiple de 4, les lignes du tableau doivent être complétées par des zéros. La donnée membre de déplacement nous sert à effectuer cette vérification et à procéder au remplissage par des zéros de toutes les lignes de la matrice lors des écritures du tableau dans le fichier. Enfin, la donnée membre de bord nous indique quelles types d'opérations nous devons mener lorsque nous essayons de lire ou d'écrire en dehors des limites de la matrice de l'image. Ces opérations sont de trois types :

1. bords nuls : l'image est entourée de valeurs nulles :
   $\forall n \in \mathbb{Z},$ si $n < 0$ ou $n > longueur, I(n, y) = 0$
   $\forall m \in \mathbb{Z},$ si $m < 0$ ou $m > largeur, I(x, m) = 0$ ;
2. bords dupliqués : les valeurs en bordure de l'image sont répétées à l'infini, c'est-à-dire :
   $\forall n \in \mathbb{Z},$ si $n < 0$ alors, $I(n, y) = I(0, y)$; si $n > longueur$ alors, $I(n, y) = I(longueur, y)$

[9] *Graphic Interchange Format*

[10] GIF a été mis au point par CompuServe en 1987 pour permettre le téléchargement d'images en couleur et souhaite désormais obtenir une redevance sur l'utilisation de son format

[11] L comme *Lempel*, Z comme *Ziv* et W comme *Welch*, le nom des trois auteurs de cet algorithme de compression

[12] compatible malheureusement uniquement à l'heure actuelle avec le format GIF87 et non GIF89

[13] voir ci-après

[14] entier compris entre 0 et 255

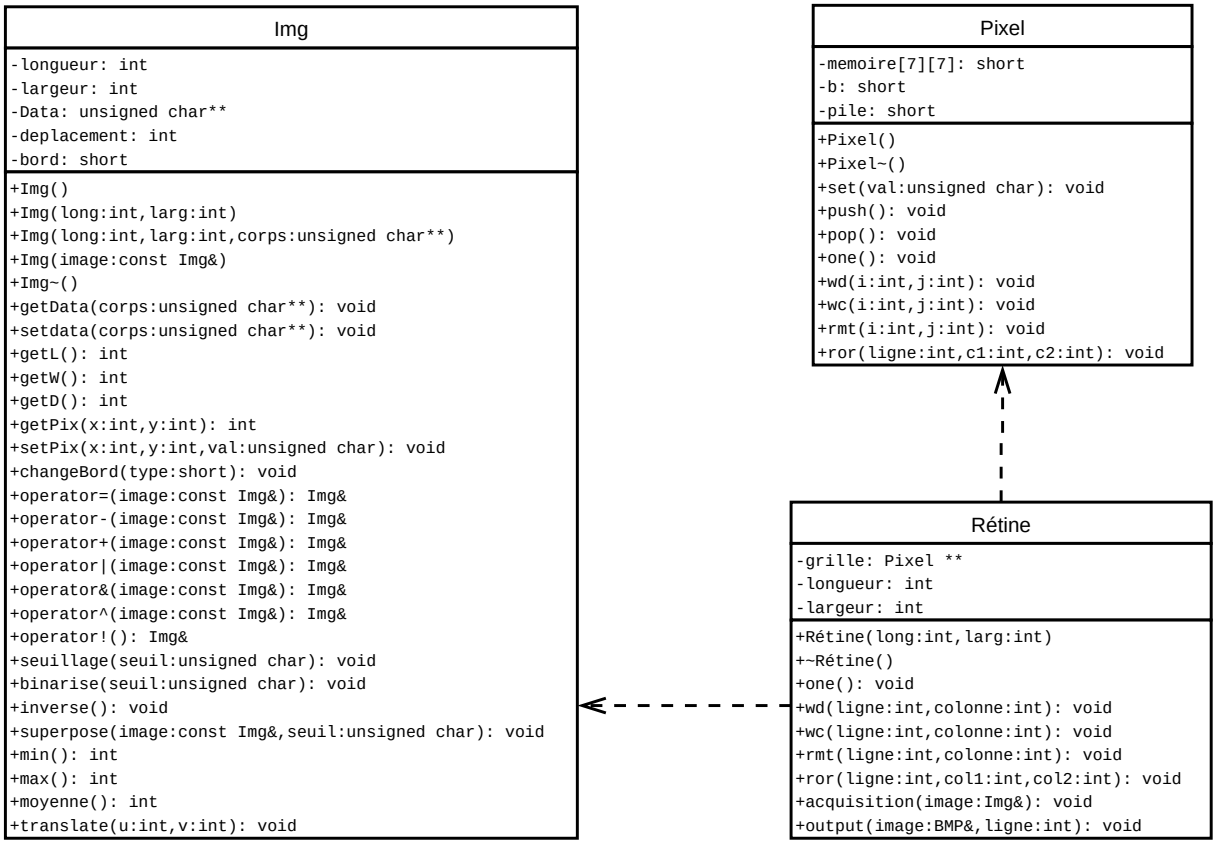



FIG. 6.1 – Schéma hiérarchique des classes utilisées dans les outils développés durant notre travail.

$\forall m \in \mathbb{Z},$ si $m < 0$ alors, $I(x,m) = I(x,0);$ si $m > largeur$ alors, $I(x,m) = I(x,largeur)$

3. bords sphériques : l'image est rendue périodique de telle sorte que :
   $\forall (x,y) \in \mathbb{Z}^2, I(x + longueur, y) = I(x, y + largeur) = I(x,y)$

La classe Img compte aussi un certain nombre de fonctions membres que l'on peut classer en trois grand types. Le premier concerne l'initialisation et la gestion des données membres. Le second sert à effectuer des opérations élémentaires sur l'image : inversion, seuillage, binarisation, recherche des valeurs maximales et minimales et translation dans les huit directions principales. Le dernier type permet de mener des opérations entre images, on trouve l'addition, la soustraction et la superposition d'images, les opérations booléennes telles que la conjonction, la disjonction et enfin l'égalité. Toutes ces fonctions nous permettent de manipuler les objets Img comme de simples variables.

Ainsi la classe Img simplifie grandement l'écriture d'algorithmes de traitement d'images en compactant le code et en le rendant aisément lisible. La figure 6.1 nous donne un aperçu des classes employées dans les logiciels développés durant notre travail.

### 6.2.3 EmulRet : un émulateur de rétine

Le premier utilitaire présenté est un émulateur de rétine. Nous avons essayé de reproduire au mieux les mécanismes du circuit rétinien afin de tester les algorithmes et d'étudier leurs comportements (l'évolution des valeurs des variables sur la rétine).

Ce logiciel est constitué d'un objet *Rétine* composé d'une grille d'objets *Pixel* de la même taille que la matrice imageuse. En entrée l'objet *Rétine* "acquiert" une image en chargeant chaque objet *Pixel* avec la valeur du niveau de gris de l'image correspondante. Puis l'algorithme est exécuté et tracé au cours du traitement, soit de manière globale en visualisant l'évolution de la valeur des variables (sous la forme de l'image traitée stockée sur une ligne de la mémoire), soit de manière locale en affichant le contenu de la mémoire d'un objet *Pixel*.

Nous reproduisons ici des mécanismes d'une architecture massivement parallèle. Ainsi pour chaque instruction rétine émulée, nous effectuons un parcours complet de tous les objets *Pixel* de l'objet *Rétine*. Nous ne pouvons pas dans ces conditions valider efficacement des algorithmes trop complexes mais de petits morceaux de ceux-ci, correspondant à des routines de l'algorithme.

En effet, pour une image $200 \times 200$ codée sur 8 bits correspondant à la matrice imageuse

de la rétine numérique Pvlsar34, nous devons effectuer $200 \times 200 \times 8 \times 6 = 1920000$ opérations par instruction en "niveau de gris", sachant que le parcours de la rétine et des registres mémoires des pixels se fait de manière séquentielle. Si l'on ajoute à cela qu'une instruction correspond en fait à un nombre variant entre une et trois opérations et qu'une simple addition représente déjà une vingtaine d'instructions rétine, on comprend pourquoi il est difficile, même pour les ordinateurs actuels les plus rapides, de reproduire un algorithme entier en temps réel.

Afin de valider de plus gros algorithmes, nous procédons par une étude limitée dans un premier temps à un seul pixel. Le nombre d'opérations étant ainsi réduit, nous pouvons tracer très rapidement l'évolution du contenu des registres mémoires de notre Pixel. Une fois l'algorithme validé sur le Pixel, nous pouvons alors passer à une simulation sur la grille entière et suivre cette fois-ci le comportement global de la rétine. Pour des traitements spatiaux, le principe reste le même à ceci près que nous démarrons notre analyse avec un voisinage (5 ou 9 pixels en fonction de la connexité) mais suivons uniquement l'évolution du pixel central (on prend alors des cas dont les résultats sont connus).

Nous ne détectons pas les mêmes erreurs lors de ces deux étapes. Le niveau Pixel permet de détecter les erreurs d'écritures du code rétinien et le niveau Rétine de relever plutôt les erreurs de conception des algorithmes. Nous avons aussi eu à faire à des phénomènes qui ne sont pas des erreurs algorithmiques mais des situations liées à l'architecture même de la rétine. Un exemple de ces phénomènes est le problème rencontré lors de l'élaboration de la technique de détection du mouvement par moyenne récursive où nous avons dû gérer l'accumulation des erreurs d'arrondis lors de calcul avec des nombres négatifs.

La Figure 6.2 présente l'interface utilisateur de l'émulateur. La seule amélioration qui reste encore à apporter à ce logiciel afin de permettre une augmentation sensible de ses performances et éventuellement de se rapprocher de traitements temps réels est l'utilisation du **multithreading**[15]. Cependant, la mise en œuvre de cette technologie est trop coûteuse par rapport à nos besoins de test et de validation au vu du nombre de tâches à paralléliser pour émuler convenablement une rétine. Nous avons donc préféré nous contenter d'une émulation partielle mais satisfaisante afin de tester des fonctions élémentaires.

Afin de rendre les tests de routines plus agréables à évaluer sur l'émulateur nous avons utilisé l'analyseur/parseur **lex/yacc**[16]. Ce couple de logiciels nous permet de traduire directement un code rétine en instruction C++ utilisable par notre émulateur de façon

[15] écriture particulière de code source permettant la parallélisation des données et donc une forme de multitâche

[16] Lex et yacc sont des outils de génération d'analyseurs lexicaux (Lex) et syntaxiques (Yacc) en langage C. "Yacc" est l'acronyme de Yet Another Compiler Compiler

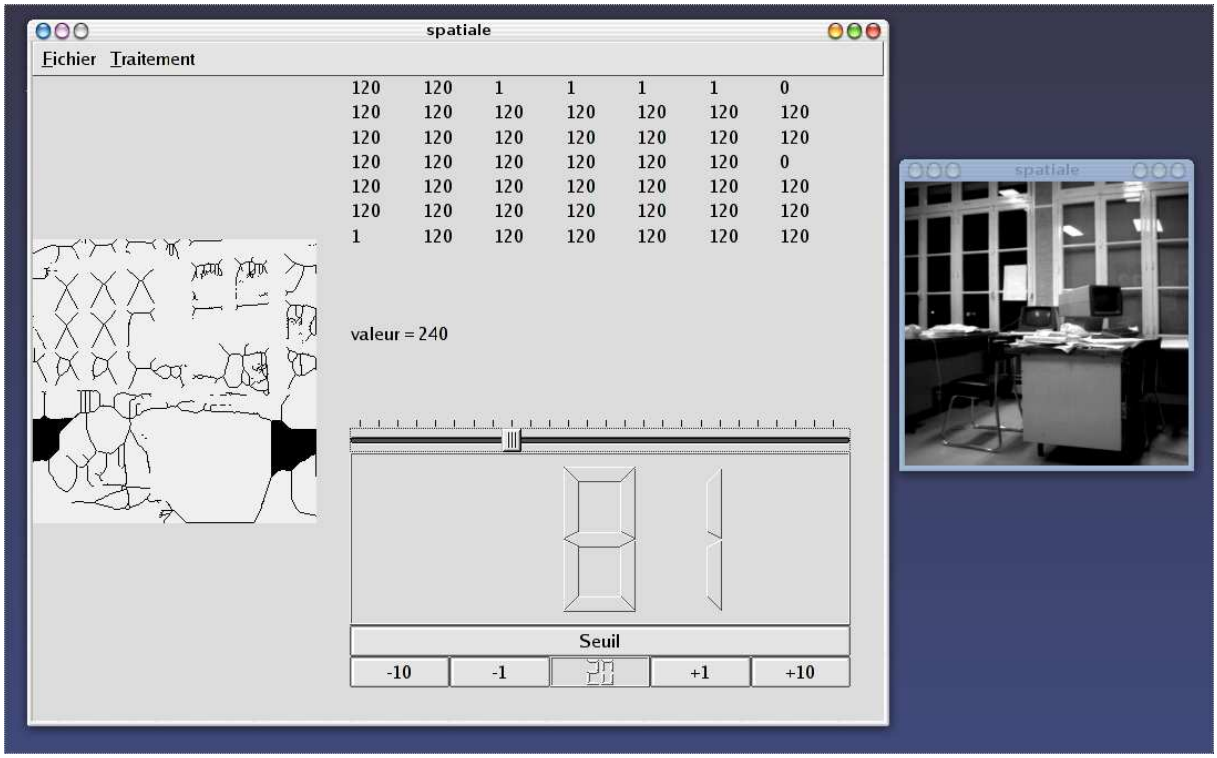



FIG. 6.2 – L'interface graphique de l'outil d'émulation EmulRet. La fenêtre de droite affiche l'image originale acquise, la fenêtre principale est découpée en trois zones : le résultat global de l'algorithme sur l'image, une régle et un compteur pour le seuillage et le nombre d'itération des algorithmes et le contenu de la grille mémoire du pixel choisi.

transparente. En effet, le premier logiciel analyse le code rétinien à la recherche de mots clés, le second fournit les règles grammaticales nécessaires pour produire un code C++ équivalent au code rétinien originel, interprétable alors par le logiciel de détection du mouvement, par exemple.

### 6.2.4 TraitLib : conception de code rétine statique

Le second outil qui est en fait le premier chronologiquement a avoir été conçu, est un logiciel générique de traitement d'images statiques. Il nous a permis de tester les algorithmes de traitement d'images classiques et de mieux appréhender les opérateurs de la morphologie mathématique.

Les techniques de conception des algorithmes du logiciel sont orientées pour être facilement utilisées sur la rétine programmable. Néanmoins, cet outil, TraitLib, se situe à un niveau conceptuel plus élevé. Il comporte en particulier une implémentation de la majorité des outils de la morphologie mathématique binaire et en niveaux de gris. Il inclut aussi toutes les fonctions dont nous avons étudié l'implémentation sur la rétine. Il s'agit de filtrage linéaire (filtre gaussien, laplacien, Sobel, Prewitt, Kirch, Roberts, ...), ainsi que les principales techniques de calcul de squelettes morphologiques binaires et en niveaux de gris. Nous avons de plus ajouté des méthodes de calcul de points d'intérêt dont les techniques morphologiques, notamment notre algorithme de calcul de points dominants morphologiques.

Cet outil présente en outre l'avantage de permettre l'étude d'un algorithme conçu pour la rétine sur une architecture classique. Nous avons ainsi pu constater notamment pour les techniques employant la morphologie mathématique oublieuse qu'elles se prêtaient fort bien à ces architectures. Ce qui nous a aussi permis d'envisager de nouvelles méthodes générales de filtrage et de détection du mouvement dans un cadre plus générale que celui de la rétine artificielle [17].

La Figure 6.3 présente l'interface utilisateur du logiciel TraitLib. Avec cet outil nous avons pu comparer et situer notre méthode de calcul de points dominants morphologiques par rapport aux autres techniques de calcul de points d'intérêt (détecteur de Harris, ...). C'est aussi pour ce logiciel que nous avons développé la plupart de nos bibliothèques de fonctions (morfo, filtre, movi, skel, nommées en fonction du type d'algorithme qu'elles contiennent) qui regroupent les traitements de type filtrage, les outils de morphologie

[17] voir A. Manzanera et J. Richefeu, "*A new motion detection algorithm based on* Σ-Δ *background estimation*", Pattern Recognition Letters 2006

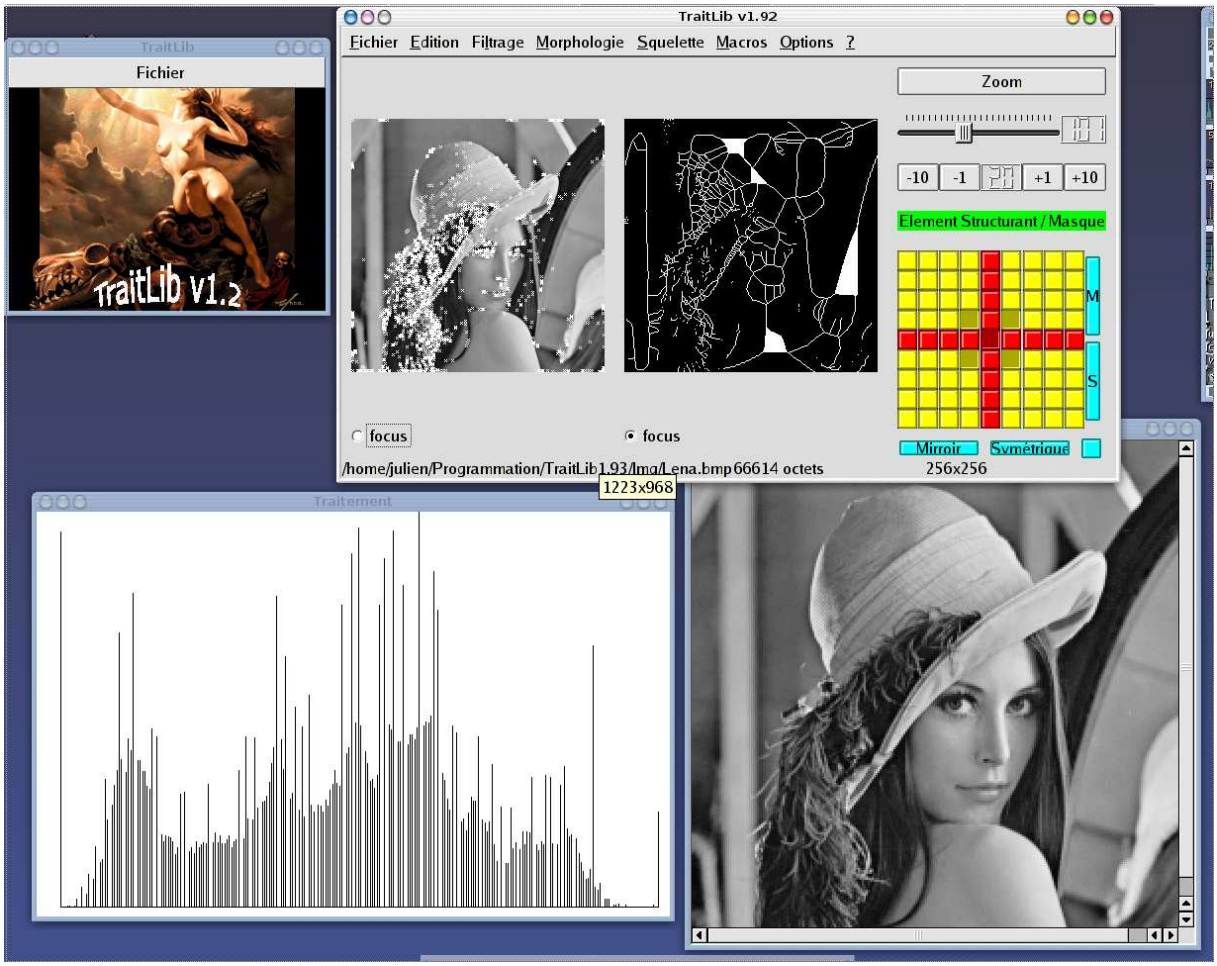



FIG. 6.3 – L'interface graphique du concepteur TraitLib. Le logiciel est agencé en fenêtres multiples : une permet de lancer choisir une image sur le disque, la feneêtre principale permet de lancer les algorithmes et de fixer les éventuelles variables (valeur de seuil, nombre d'itération, noyau morphologique, etc...), une autre affiche un zoom de l'image choisie et la dernière affiche un histogramme de l'image désirée.

mathématique, ceux de squelettisation et de calcul de points d'intérêt et enfin les fonctions permettant la détection du mouvement dans une séquence vidéo. A ceux-là, nous pouvons ajouter une bibliothèque d'outils de conversion d'image et de fonctions utiles au traitement des fichiers informatiques contenant des images.

### 6.2.5 TraitMotion : conception d'algorithmes de détection du mouvement pour rétines artificielles

Le logiciel suivant proposé est principalement un outil de détection du mouvement dans une séquence d'images : TraitMotion. Il permet de tester des techniques de détection d'objets mobiles sur une vidéo présente sur le disque dur de l'ordinateur.

Nous avons ajouté par la suite un module nous permettant l'interfaçage de l'outil avec une webcam via le port USB. Cette fonctionnalité n'a été que peu utilisée car elle ne se comporte pas de la même manière selon le système d'exploitation en fonction de la façon dont les pilotes de la webcam sont documentés.

En effet, sous Windows, la détection depuis le flux provenant de la webcam ne pose aucun problème, la vidéo étant simplement découpée en succession d'images au format JPEG, écrites ponctuellement sur le disque dur puis traitées de façon habituelle par le logiciel. Cependant, cette méthode est dépendante des fréquences de lecture-écriture, rendant le traitement provenant d'une webcam relativement lent. Sous Linux, nous ouvrons un port USB en lecture puis récupérons les images sous la forme de paquets de données brutes pré-formatées. Ainsi, le débit de traitement est très bon et les calculs effectués en temps quasi réel. Cependant, la webcam utilisée n'est pas supportée sous Linux et il nous a fallu utiliser des pilotes provenant de développeur amateur[18] qui ne nous permettent pas d'obtenir un débit et une taille d'image comparable à celle obtenue sous Windows[19]. Enfin, cette même webcam n'est pas reconnue sous Mac OS X, nous n'avons donc pas pu tester le traitement provenant d'une webcam sous ce système.

La Figure 6.4 présente l'interface utilisateur du logiciel TraitMotion. Nous avons pu tester et améliorer toutes nos techniques de détection du mouvement grâce à cette interface et les comparer aux autres techniques étudiées lors de la thèse. Outre nos propres méthodes, les algorithmes que compte le logiciel sont :

– les techniques par différence d'images (différence inter-trames, différence à un fond, différence robuste [Bel96], ...) ;

[18] `http://www.smcc.demon.nl/webcam/`

[19] $640 \times 480$ en 40 fps sous Windows contre seulement $320 \times 200$ en 15 fps sous Linux

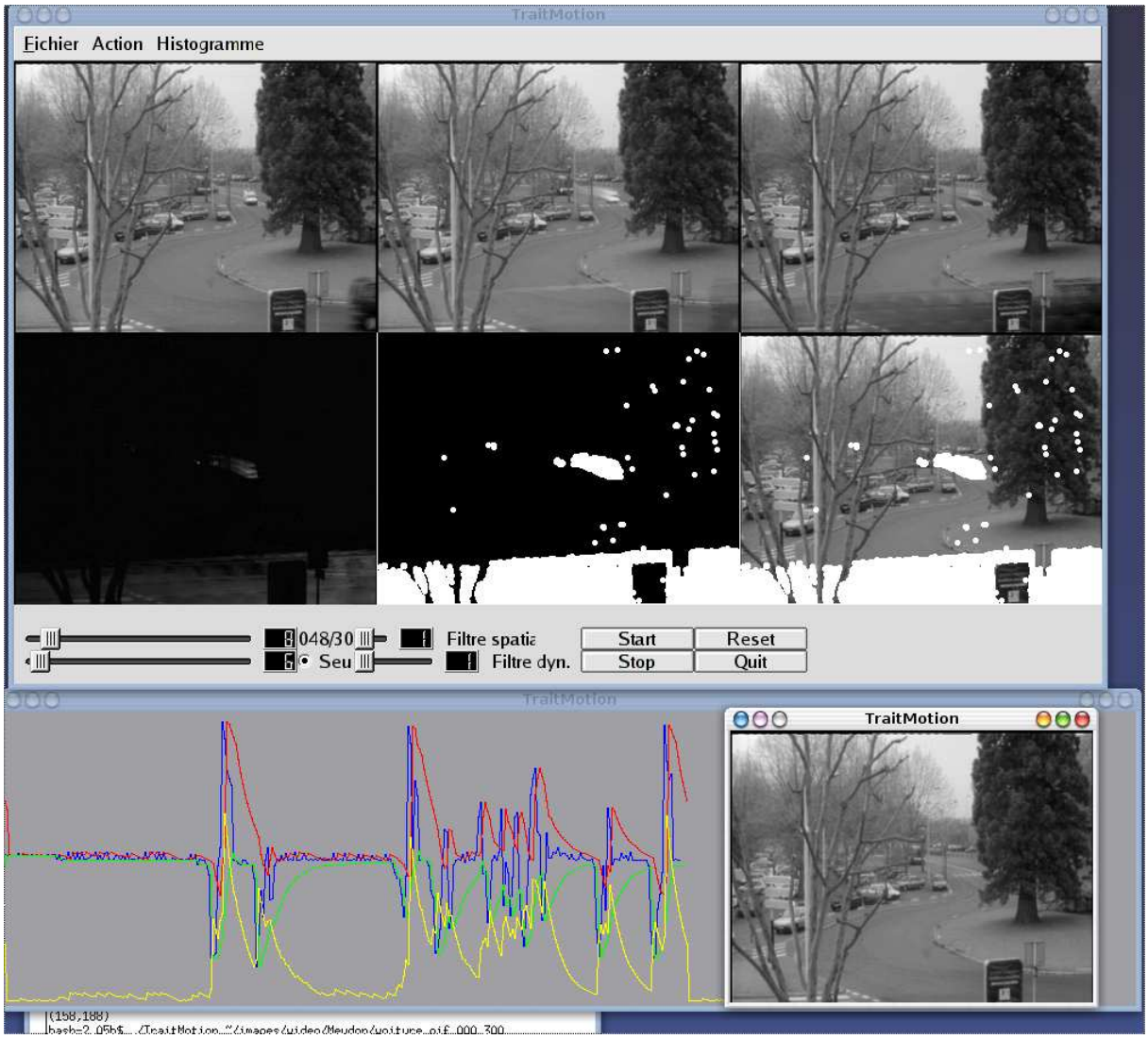



FIG. 6.4 – L'interface graphique de l'outil de validation d'algorithme de détection du mouvement TraitMotion. La fenêtre principale affichage 6 images correspondantes en fonction des algorithmes à l'image originale à l'instant $t$ et jusqu'à 5 images traitées. Une fenêtre affiche l'image originale à $t = 0$ afin de sélectionner un pixel à suivre dans un graphique (dernière fenêtre) .

– les techniques de filtrage (Sobel, Prewit, gradient, laplacien, Roberts 3D, ...) ;
– les techniques de moyennes (moyenne, moyenne récursive, ...) ;
– les techniques morphologiques (tampon morphologique simple et mixte, gradient morphologique [Agn04], ...).

### 6.2.6 Logiciel de calcul de corrélation

A la fin de notre travail de recherche, nous avons mis au point un dernier logiciel nous permettant de tester nos algorithmes de mise en correspondance de points dominants morphologiques sur des couples d'images. Ce logiciel est moins élaboré que ceux présentés auparavant et nécessiterait quelques ajouts de fonctionnalités afin d'être plus agréable à utiliser (travail sur une séquence vidéo, modification des paramètres, etc...) Cependant, ce petit outil nous a permis de valider l'algorithme présenté Chapitre 5 avant d'envisager son implémentation rétine.

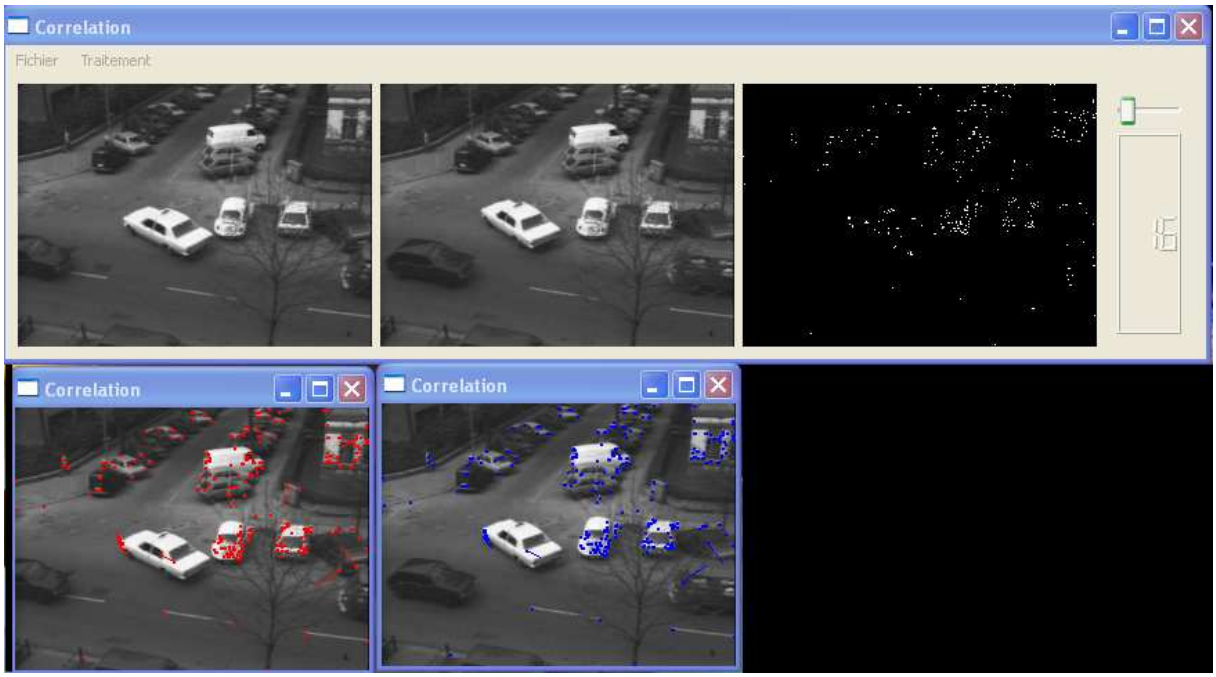



FIG. 6.5 – L'interface graphique de l'outil de validation de l'algorithme de mise en correspondance de points dominants morphologiques.

La Figure 6.5 présente l'interface utilisateur du logiciel de validation de l'algorithme de mise en correspondance de points dominants morphologiques. Ce logiciel utilise les mêmes

bibliothèques de fonctions que celles présentées auparavant pour les autres logiciels.

## 6.3 Aide à la génération de code

Cette seconde partie présente les optimisations apportées aux implémentations des algorithmes sur le circuit rétinien. Ces améliorations sont principalement de trois types :

1. la réduction du code généré ;
2. l'aide à la programmation ;
3. une allocation intelligente de la mémoire dans le pixel rétinien.

Afin de parvenir à ces améliorations, la méthodologie consiste en trois points :

1. Partitionnement de la mémoire ;
2. Ecriture des codes des routines avec des registres **génériques** ;
3. Allocation des registres de données.

### 6.3.1 Réduction du code généré

La première grande optimisation vient du fait de l'agencement même du code rétinien. En effet, comme nous l'avons expliqué dans les sections précédentes, nous effectuons beaucoup plus d'opérations d'écriture que de lecture (deux fois plus en moyenne). Nous rappelons que chaque lecture est au moins suivie d'une écriture afin de rafraîchir la donnée lue (sauf bien entendu dans le cas de lecture avec perte). De plus, les écritures coûtent plus chers que les lectures en nombre d'instructions séquencées par le Nios (3 pour les lectures contre 6 pour les écritures). Nous avons donc cherché à rassembler des portions de code afin d'effectuer des suites d'écritures et d'éviter ainsi des lectures-écritures supplémentaires. Un exemple de ce type d'optimisation est illustré dans le Chapitre 1 lors de l'amélioration des fonctions de soustraction et d'addition.

Le code à exécuter sur la rétine se trouve sur une mémoire RAM double port partagée entre le Nios et un FPGA sur la carte de contrôle de la rétine (voir Chapitre 1 Section 1.2.3 pour plus de détails). Or, afin de permettre à des algorithmes plus complexes (et donc des codes plus longs) d'être fonctionnels nous avons réduit la zone mémoire réservée aux variables. Cela implique un contrôle strict du nombre de variables pour éviter les problèmes d'écritures sur des zones de mémoires non allouées qui provoqueraient irrémédiablement

un plantage du système embarqué sur le processeur ARM[20].

Dans l'optique d'utiliser le moins de mémoire possible, nous n'utilisons que des variables locales sauf cas exceptionnel. En effet, la zone mémoire allouée aux variables locales ainsi que la taille du code sont bien plus réduits que dans le cas de variables globales. Ceci est en partie du à l'optimisation faite par le compilateur spécifique compatible avec le Nios. En effet, le compilateur est capable de réallouer de façon dynamique des variables locales inutilisées.

### 6.3.2 Aide à la programmation

Dans cette section nous présentons des méthodes d'aide à la programmation exploitant les principes de généricité et d'orthogonalité[21] du code applicable à la rétine.

Dans la mesure où l'on a chiffré le nombre d'instructions nécessaires pour l'appel de fonction, nous essayons, partout où cela est possible, de remplacer des bouts de code qui sont répétés plusieurs fois par des fonctions. En effet, en règle général, à partir de quelques opérations (entre 3 et 5 en fonction du nombre de lectures et d'écritures), il est moins coûteux d'utiliser une fonction, en comptant les instructions nécessaires à son appel, plutôt que de laisser les lignes de codes répétées.

Mieux encore, nous nous efforçons de remplacer ces fonctions par des macro interprétées par le précompilateur, moins coûteuses en taille de code. A ce niveau d'optimisation, il est en effet nécessaire d'écrire des routines optimisées que le précompilateur peut interpréter. Le compilateur du Nios intégre en effet une gestion d'optimisation du code qui permet de le rendre plus compacte lorsque l'on utilise des macros définies lors de la précompilation plutôt que des fonctions classiques (optimisation en terme de taille du code source).

La dernière étape d'optimisation est l'écriture de routines génériques. Nous avons intégré aux macros précédemment introduites une technique nous permettant d'écrire des boucles de calculs sur les bits lorsque l'algorithme nous le permettait. Nous avons ainsi la possibilité d'écrire une addition entre deux mots (entre tous les bits du mot) sur la rétine

[20]L'architecture ARM était initialement destinée à un ordinateur de la société *Acorn*, puis elle a été complétée pour devenir une offre indépendante pour le marché de l'électronique embarquée.

[21]Le jeu d'instructions d'un ordinateur est dit orthogonal lorsque toutes les instructions ou presque peuvent s'appliquer à tous les types de données. Un jeu d'instruction orthogonal simplifie la tâche du compilateur puisqu'il y a moins de cas particuliers à traiter : les opérations peuvent être appliquées telles quelles à n'importe quel type de donne

en une seule instruction.

Pour comparaison, voici un bout de code correspondant à la soustraction entre les mots Img et Cor, le résultat est écrit en Diff. La fonction *Soustraction-2k-Bit()* effectue une soustraction binaire en prenant le registre (y6,x1) comme retenue et (y6,x2) et (y6,x3) comme opérandes. Le résultat se trouve en (y6,x2). L'exemple présenté est la portion de code pour le calcul du premier bit du mot :

```
//Difference entre Img et Cor
//S=0
one; wc y6 x1;

//Bit n0
//Copie de Img_0 en y6;x2
rmt1 y1 x0; wd y1 x0;
wd y6 x2;
//Copie de Cor_0 en y6;x3
rmt1 y2 x0; wd y2 x0;
wd y6 x3;

Soustraction-2k-Bit();

//Mise a jour de Diff_0
rmt1 y6 x2;
wd y3 x0;

...
```

Le code suivant réalise la même opération avec les mêmes variables mais présente le code pour le mot entier de 8 bits :

```
//Difference entre Img et Cor
//S=0
one; wc IO_1;

for (nbit=0; nbit<8; nbit++)
SOUSTRACTION(Img[nbit],Cor[nbit],Diff[nbit]);
```

Le gain est donc dans cet exemple de 3 instructions rétines au lieu de $8 * 8 + 2 = 66$

(8 instructions pour 8 bits plus l'initialisation) dans la version initiale du code.

### 6.3.3 Allocation mémoire

La règle la plus élémentaire de simplification de l'écriture du code provient de l'utilisation d'une ligne entière dans la matrice de registre mémoire dédiée aux calculs dans les fonctions.

De la même manière nous avons découpé la grille de mémoire présente en chaque processeur en différentes parties afin de rendre l'allocation de cette mémoire plus simple voire même de la rendre automatique. Cette division n'est bien sûr que virtuelle, nous ne disposons pas de mécanisme électronique ou algorithmique pour véritablement allouer de la mémoire à une variable comme le ferait le langage *C* par exemple. Une ligne est dédiée au calcul, une à l'image acquise, une ou plus pour les résultats (en fonction des algorithmes) et quelques registres pour les variables temporaires. A cela, il faut ajouter les registres dédiés au communications entre pixels.

Dans le même souci de simplicité nous avons opté pour un prototypage identique pour la plupart de nos fonctions. Ainsi, les mêmes registres mémoires sont utilisés en entrées et en sorties pour toutes les fonctions élémentaires. De même, quelques registres supplémentaires sont réservés afin de stocker les variables temporaires dont nous avons éventuellement besoin.

## 6.4 Conclusion

Durant nos trois années de recherche et de développement d'algorithmes sur système de vision à base de rétine numérique, nous avons développé une bibliothèque importante de fonctions élémentaires de traitement d'images dans un premier temps (TraitLib), de détection du mouvement (TraitMotion) et enfin d'opérateurs d'estimation du mouvement (corrélation). Parmi ces outils, on compte des fonctions de détection de points isolés, de points dominants, de calcul de squelette, de fonctions permettant la poursuite de cibles dans des trames successives, ....

Nous avons aussi utilisé un certain nombre de techniques permettant à l'utilisateur d'écrire un code plus simplement. L'objectif final est de n'avoir plus qu'un code lisible par un développeur quelconque à la place du pseudo-assembleur rétinien.

Cependant, il reste un travail important à un développeur pour "rentrer" dans le concept rétine et pour concevoir de nouvelles fonctionnalités. Bien que tout soit en place pour faciliter le travail du programmeur par la définition des macros, des fonctions élémentaires optimisées, d'outils de tests et de validation, il manque tout de même les liens entre ces fonctionnalités.

En effet, idéalement on souhaiterait écrire des algorithmes dans un langage de haut-niveau comme le $C++$ et disposer directement du code rétinien à exécuter en sortie. Les spécificités de la rétine, notamment ses capacités mémoires réduites nous amènent, pour des algorithmes complexes, à intervenir au plus près du circuit en ce qui concerne l'agencement et l'optimisation du code. Les outils dont nous disposons actuellement, bien qu'imparfaits, sont un compromis acceptable pour un développeur qui devra néanmoins s'accommoder d'une phase d'apprentissage encore nécessaire aujourd'hui.

# Conclusion et perspectives

L'objectif de cette thèse est de montrer que des Rétines Artificielles Numériques Programmables (RANP) permettent d'effectuer des traitements de bas et moyen niveaux adaptés à un système de détection et d'analyse du mouvement.

Dans une première étape, nous avons rappelé la problématique de la vision et du traitement d'image pour la perception du mouvement. Nous avons décrit de quelle manière le règne animal remplit cette fonction essentielle.

Or, si les fonctions de perception visuelle des systèmes de vision biologique semblent si efficaces qu'elles ne demandent aucun processus conscient, cette problématique reste encore opaque pour les chercheurs. Il n'est cependant pas interdit de s'inspirer partiellement des solutions utilisées par la nature pour tenter d'en exploiter certaines caractéristiques. C'est dans cette optique que les RANP, sans reproduire les processus du vivant, reprennent à leur avantage l'organisation massivement parallèle et les traitements essentiellement locaux pour implanter aussi efficacement que possible des traitements à des fins de perception.

L'état de l'art réalisé dans le premier chapitre tente de présenter les diverses solutions adoptées pour implanter des traitements au sein du circuit rétinien. Nous avons ainsi mis en évidence les avantages qu'offrent les circuits numériques sur leurs homologues analogiques. Ce Chapitre se termine par une présentation des projets de recherche qui ont abouti à l'élaboration de la rétine numérique artificielle.

Les traitements adaptés au circuit rétinien ont été présentés dans le Chapitre 2 dont les outils de la morphologie mathématique puis les fonctions élémentaires indispensables à l'élaboration d'algorithmes plus complexes. Quelques exemples de fonctions élémentaires ont été ensuite présentés en détails (addition, soustraction, valeur absolue, comparateur, détecteurs de points isolés binaires, détecteur de contours morphologiques, opérateur point extrême binaire), notamment en ce qui concerne le code "rétine" utilisé. une partie des

fonctions élémentaires morphologiques sont rappelées en **??**.

Notre réflexion a débuté au Chapitre 3 par l'identification des nombreuses contraintes qui s'imposent dans le cadre de la détection du mouvement dans des scènes complexes par un capteur fixe. Après avoir survolé les différentes techniques proposées dans la littérature, nous avons étudié séparément trois méthodes élaborées au cours de notre étude.

La première méthode présentée, basée sur le calcul d'une moyenne récursive, est peu coûteuse en temps de calcul et en ressource mémoire. Elle est aussi moins adaptative aux conditions de la scène, elle est moins précise en terme de localisation et enfin elle est sujette aux phénomènes de fantôme. Deux raffinements à cette méthode ont ensuite été proposées, une technique de rebouclage et un système de seuillage dynamique, nous permettant d'obtenir de bien meilleurs résultats à moindre frais.

La seconde méthode est une nouvelle technique récursive basée sur des filtres hybrides. Les opérateurs de morphologie oublieux offrent une plus grande capacité de détection mais sont moins précis dans la localisation des objets détectés. De la même manière que nous l'avons fait pour la moyenne récursive, un seuillage dynamique a ensuite été utilisé avec succès.

La dernière est une méthode statistique basée sur l'estimation $\Sigma$-$\Delta$, une analogie avec le convertisseur analogique / numérique en électronique. Elle est plus adaptative localement et temporellement. La localisation est plus précise mais elle est plus sensible aux modifications brutales du fond. Un système de filtrage spatio-temporelle basé sur des reconstructions géodésiques permet d'améliorer les résultats de la détection. Quelques raffinements ont été ensuite apportés a cet algorithme, notamment un paramétrage automatique du terme oublieux, l'apport de filtrages spatio-temporels de l'estimation. Il a été aussi montré que nous pouvons utilisé le filre $\Sigma$-$\Delta$ conjoitement à une technique de régularisation Markovienne ou encore d'une estimation multimodale du fond.

Nous nous sommes ensuite attachés au Chapitre 4 à décrire des primitives de calculs permettant l'estimation du mouvement. Nous avons alors introduit les opérateurs d'estimation des structures saillantes spatiales 2D, les points dominants morphologiques et les squelettes.

Les points d'intérêt sont décrits comme des points de fixation visuels construits à l'aide de squelettes morphologiques particuliers. Nous avons tenté de généraliser cette notion en effectuant le calcul du squelette directement en niveaux de gris mais les résultats obtenus n'ont pas été satisfaisants dans notre contexte.

Au Chapitre 5, nous avons utilisé le calcul des points dominants dans le cadre de

l'estimation du mouvement. Nous avons alors effectué la mise en correspondance des structures saillantes avec une méthode simple de corrélation. Nous nous sommes confrontés aux capacités limitées de la rétine en terme de mémorisation et de traitement. Nous avons alors adapté les algorithmes de corrélation aux propriétés de circuit afin d'obtenir les résultats escomptés. Il semble toutefois qu'à ce stade, le choix de poursuivre les calculs de corrélation sur la rétine ou de le déporter sur un système externe se pose. Vient alors le problème du choix de la représentation des données en sortie du circuit. Une étude du type de données et de la manière de les formater reste à être menée.

Suite aux différents problèmes liés à l'élaboration d'algorithmes sur le circuit rétinien soulevés au cours de notre étude, nous avons présenté une méthodologie de conception sur système à base de rétine numérique. Nous avons ainsi introduit les différents logiciels d'aide à la conception et à la validation d'algorithmes. Ces algorithmes ne sont pas spécifiquement dédiés à la rétine artificielle mais sont écrits dans un style orienté pour ce circuit. C'est ainsi que nous avons pu développer les méthodes de détection du mouvement que nous avons proposées indépendamment de l'architecture rétinienne[22], et qui ont été implantées sur d'autres processeurs (x86/SSE2, PowerPC/Altivec, Maille associative d'Orsay) [DMLM05].

En conclusion, nous formulerons certaines remarques pour enrichir notre réflexion sur l'étude du mouvement sur le système de vision rétinien. Tout d'abord, comme cela a été relevé plusieurs fois au cours de notre étude, l'architecture actuelle de la rétine fonctionne en mode **synchrone**, ce qui limite la portée de certains calculs. En effet, certaines opérations simples (calcul de minimum local, calcul d'un arbre couvrant ou tout autre calcul sur une région) demandent des ressources de calcul très importantes ou ne sont tout simplement pas effectuables sur notre modèle de rétine. Or, une étude architecturale a été menée conjointement à nos travaux par Valentin Gies [Gie05] dans l'optique de l'élaboration d'une rétine numérique **asynchrone**. Il apparaît qu'un tel circuit fournirait une portée plus régionale aux opérations et aurait donc un champ d'application plus étendu. On compte entre autres la segmentation, le calcul de contour, de ligne de partage des eaux, etc... L'idéal serait de disposer d'un circuit mixte synchrone / asynchrone pour profiter des avantages des deux modes de fonctionnement.

22

- A. Manzanera et J. Richefeu, "*A robust and computationally efficient motion detection algorithm based on Σ-Δ background estimation*", ICVGIP, Kolkata, India Décembre, 2004.
- J. Richefeu et A. Manzanera, "*A new hybrid differential filter for motion detection*", ICCVG, Warsaw, Poland, Octobre, 2004.
- J. Richefeu et A. Manzanera, "*Morphological dominant points detection for motion analysis on programmable retina*", ISSPA, Paris, France, Juillet, 2003.

D'un point de vue architectural, la rétine comporte une carte de contrôle disposant d'un couple ARM-FPGA très sous exploité. Le système qui pilote cette carte est rudimentaire et offre des capacités limitées notamment en terme de communication. Afin d'avoir une plus grande liberté d'action sur le circuit et d'améliorer les performances, il serait intéressant d'installer un système d'exploitation Linux embarqué dans un module de mémoire externe ou mieux, sur une mémoire flash installée sur la carte. Les systèmes embarqués offrent en effet actuellement un nombre grandissant de projets et il existe donc une grande expérience pour ce type d'application (utilisation du couple ARM-FPGA, réseau, ...). On disposerait alors d'outils performants permettant d'optimiser nombre de fonctions (communication, débit, ...).

Deux autres champs de recherche qui n'ont pas encore été étudiés mais qui offrent cependant des possibilités intéressantes sur un système comme celui de la rétine numérique sont l'implantation de **réseaux de neurones**[23] directement sur le circuit, et d'**algorithmes génétiques**[24]

L'usage des réseaux de neurones a déjà fait l'objet de nombreuses études sous la forme des circuits de réseaux de neurones cellulaires (Chapitre 1, Sous-Section 1.2.2). Il serait donc intéressant de tenter d'appliquer les résultats obtenus avec ces circuits sur la rétine. Cela demande cependant de repenser les calculs, le plus souvent discret, afin de les porter sur le circuit numérique disposant de quelques bits par pixels. En effet, ici, ce sont les pixels qui représentent les neurones, et donc le codage des fonctions, des poids et des variables se fait au sein de la mémoire du pixel. Plusieurs applications sont envisageables dans le domaine du mouvement, on pense tout d'abord à l'estimation et à l'utilisation des réseaux de neurones comme autant de particules comportant leurs propres informations

[23]Un réseau de neurone est un groupe interconnecté de neurones. Chaque neurone est une fonction mathématique à une ou plusieurs entrées et une ou plusieurs sorties et dispose également éventuellement de poids sur chacune de ses entrées. Le réseau peut comporter plusieurs couches mais seules les couches en entrée et en sortie sont accessibles, les autres sont appelés couches cachées. Une fois le calcul initié (en donnant les valeurs à la couche d'entrée), plusieurs itération sont parfois utiles avant d'obtenir un résultat, on a alors généralement un rebouclage entre la couche de sortie et la couche d'entrée. Les réseaux de neurones sont ainsi utilisées dans des problèmes afin de simuler un comportement de raisonnement ou l'on converge vers une solution mais où les étapes intermédiaires peuvent prendre des valeurs quelconques.[Ros62]

[24]les algorithmes génétiques est une technique de recherche employée en informatique pour trouver les solutions approximatives à l'optimisation et pour la recherche de problèmes. Les algorithmes génétiques sont une classe particulière d'algorithmes évolutionnaires qui emploient des opérateurs inspirées par l'évolution biologie.

Des algorithmes évolutionnaires sont typiquement mis en application lors de simulation sur ordinateur dans laquelle une population de représentations abstraites (appelées chromosomes) de candidats (appelées individus) à un problème d'optimisation évolue vers de meilleures solutions par l'application des "opérateurs génétiques" (sélection, mutation, croisement) sur plusieurs génération. Une classe particulière d'algorithmes évolutionnaires est dédiée à la conception d'algorithmes ("programmation génétique").

permettant d'évaluer le mouvement le plus probable.

Dans le même esprit, lors de notre stage de DEA effectué dans les Laboratoires d'Electronique et d'Informatique de l'ENSTA, nous avons étudié une implémentation d'un **perceptron multicouche**[25] sur la rétine afin d'effectuer une reconnaissance de chiffres issus de la détection. La couche d'entrée est la grille de pixel entière dont les valeurs en entrée sont issues de l'acquisition de l'image brute, traitées afin d'être centrées sur le chiffre à reconnaître. Chaque pixel contient la valeur du poids issue de l'apprentissage menée précédemment. Il a été montré qu'un tel algorithme permet d'obtenir de très bon résultats après une phase d'apprentissage selon la règle du WIDROW-HOFF. Cependant, il est difficilement envisageable de coder un réseau trop important sur la rétine. En effet, les poids des noeuds du réseau sont typiquement codés par des flottants et consomment alors un nombre important de registres mémoires. Il nous faudrait alors mettre en place un mécanisme afin d'adapter un tel réseau sur le circuit rétinien[26].

Le principe des algorithmes génétiques nous permet de faire évoluer une population de programmes sur plusieurs générations en les croisant, les divisant jusqu'à atteindre un certain maximum d'efficacité qui serait trop coûteux à trouver par une autre technique. On peut reproduire ce mécanisme de sélection avec le code rétinien chargé sur le circuit et atteindre en quelques cycles un algorithme stabilisé. Le travail du développeur en serait ainsi grandement facilité, le niveau du langage du code rétine étant suffisamment proche de l'assembleur pour nous permettre de mettre en place un tel mécanisme.

Le projet de la rétine artificielle numérique programmable s'inscrit dans un mouvement initié par de nombreuses équipes de recherche et d'entreprise cherchant à intégrer des modules électroniques performant dans un environnement de travail varié. Cette recherche d'intégration s'effectue le plus souvent en accord avec les points suivants :

- miniaturisation des composants en passant notamment par de nouveaux formats ou de nouvelles matières (remplacement du silicium, utilisation de la fibre optique, ...) ;
- migration de fonctions de calculs ou d'algorithmes depuis le système vers le matériel (comme par exemple pour la compression/décompression MPEG4 en vidéo ou le calcul de forme 3D dans les cartes vidéos de dernières génération, ...) ;
- adaptation du matériel au problème pour répondre au mieux aux besoins (multithreading, multigranularité, ...)

Parmi les applications porteuses actuellement et où la rétine trouverait sa place, on compte

[25] Le perceptron est un type de réseau de neurone artificiel inventé en 1957 dans les laboratoires Cornell Aeronautical par Franck Rosenblatt. Il a été montré que le plus simple réseau de neurone sans rétroaction, comme le perceptron, est un classifieur linéaire.[Ros58]

[26] J. RICHEFEU, "*Détection / Reconnaissance d'objets sur rétines artificielles mixtes synchrones / asynchrones*", rapport de DEA, Septembre 2002

tous les systèmes embarqués utilisant le traitement d'image, qu'il soit statique ou dynamique. De la même manière, le domaine de la réalité augmentée est tout à fait propice à l'utilisation des système basés sur une rétine numérique programmable. En fait, en rappelant que les avantages de la rétine sont principalement la compacité et l'autonomie, et face à l'explosion des systèmes mobiles, on voit pourquoi ce type de système est en ce moment étudié par de nombreuses équipes de recherche à travers le monde.

# Annexe A

# Morphologie mathématique

## A.1 Morphologie mathématique binaire

??

Comme cela est montré dans le travail mené par [Gar84], des liens étroits unissent l'algorithmique d'une rétine booléenne et la **morphologie mathématique** ([Ser82] et [Ser88]). Revenons donc sur les concepts fondamentaux de cette discipline que nous utilisons abondamment dans la suite du document.

Une image binaire numérique est considérée comme une partie de $E = \mathbb{R}^N$ ou $E = \mathbb{Z}^N$.

### A.1.1 Opérations de base

**Définition 39** *$A$ et $B$ deux sous-ensembles de $\mathbb{Z}^2$. L'addition de Minkowski de $A$ et $B$ est l'ensemble :*

$$A \oplus B = \bigcup_{b \in B} A_b = \{a + b | a \in A, b \in B\} = \{x | \exists a \in A, \exists b \in B, x = a + b\}$$

*La soustraction de Minkowski de $A$ et $B$ est l'ensemble :*

$$A \ominus B = \bigcap_{x \in B} A_x = \{x | \forall y \in B, x - y \in A\} = \{z | \forall y \in B, \exists x \in A, z = x - y\}$$

**Hypothèse 1** *nous notons* $\breve{X}$ *= -X le symétrique de X par rapport à l'origine :*

$$-X = \{-x | x \in X\}$$

### A.1.2 Érosion et dilatation

**Définition 40** *L'érodée de X par B (élément structurant) est défini par la soustraction de X par* $\breve{B}$ *:*

$$E(X, B) = X \ominus \breve{B} = \{x \in E | B_x \subset X\} = \{x | \forall y \in B, x + y \in X\}$$

L'érosion est anti-extensive seulement si l'origine est dans $B$ :

$$E(X, B) \subset X$$

Croissante :

$$X \subset Y \Rightarrow E(X, B) \subset E(Y, B)$$

Distributive à droite par rapport à l'intersection et la réunion :

$$E(X, B \cup C) = E(X, B) \cup E(X, C)$$

$$E(X, B \cap C) = E(X, B) \cap E(X, C)$$

Distributive à gauche pour l'intersection :

$$E(X \cup Y, B) \subset E(X, B) \cup E(Y, B)$$

$$E(X \cap Y, B) = E(X, B) \cap E(Y, B)$$

Décomposable (pseudo-associativité) :

$$E(E(X, B), C) = E(X, B \oplus C)$$

**Définition 41** *Le dilaté de X par B est défini par l'addition de X par* $\breve{B}$ *:*

$$D(X, B) = X \oplus \breve{B} = \{x \in E | B_x \cap X \neq \emptyset\} = \{y | \exists x \in X, \exists b \in B, y = x - b\}$$

La dilatation est extensive si et seulement si le centre (l'origine) de l'élément structurant $B$ est dans $B$.

$$O \in B, X \subset D(X, B)$$

Croissante :

$$X \subset Y \Rightarrow D(X, B) \subset D(Y, B)$$

Distributive à droite (sur l'élément structurant) par rapport à l'intersection et la réunion ensemblistes :

$$D(X, B \cup C) = D(X, B) \cup D(X, C)$$

$$D(X, B \cap C) = D(X, B) \cap D(X, C)$$

Distributive à gauche pour la réunion mais non pour l'intersection :

$$D(X \cup Y, B) = D(X, B) \cup D(Y, B)$$

$$D(X \cap Y, B) \subset D(X, B) \cap D(Y, B)$$

Associative :

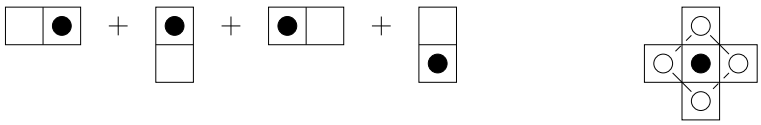
$$D(D(X, B), C) = D(X, B \oplus C)$$

**Propriété 7**

$$E(X, B) = \neg D(\neg X, B)$$

**Remarques :**

- La dilatation par un élément structurant situé à droite de son centre étend l'objet vers la gauche.
- L'addition de Minkowski avec un objet situé à droite de l'origine étend un objet vers la droite.
- Si l'élément structurant est symétrique par rapport à l'origine les deux opérations précédentes sont équivalentes.
- L'érosion et la dilatation ne sont pas inverses puisque la dilatation ne fera pas réapparaître les petites composantes connexes que l'érosion aura éliminées.



(1) (2)

FIG. A.1 – L'opérateur de rétine correspondant à l'érosion en (1) 8-connexité et en (2) 4-connexité.

Calcul de l'érosion sur la rétine

Définition de l'érosion élémentaire en 4-connexité :

$$E_4 = P \wedge P.N \wedge P.E \wedge P.S \wedge P.W$$

pseudo-code rétine :

$$\begin{aligned} P_1 &= P.E \wedge P.N \\ E_4 &= P \wedge (P_1 \wedge P_1.SW) \end{aligned}$$

La Figure A.1 présente schématiquement les opérateurs rétiniens pour l'érosion en 4- et en 8-connexité. le point noir représente l'origine, l'érosion en 8-connexité est l'association d'un ET logique du point noir avec son voisin dans les quatres directions cardinales successivement. L'érosion 4-connexe est la conjonction du point original et de ses quatres voisins directes.

Définition de l'érosion élémentaire en 8-connexes :

$$E_8 = P \wedge P.N \wedge P.NE \wedge P.E \wedge P.SE \wedge P.S \wedge P.SW \wedge P.W \wedge P.NW$$

pseudo-code rétine :

$$\begin{aligned} P_1 &= P \wedge P.E \\ P_1 &= P_1 \wedge P_1.S \\ P_1 &= P_1 \wedge P_1.W \\ E_8 &= P_1 \wedge P_1.N \end{aligned}$$

L'érosion 4-connexe nécessite donc 3 opérations et la version 8-connexe 4 opérations.

### A.1.3 Gradient morphologique

Le grdient morphologique est défini par la différence entre le dilaté et l'érodé de l'image.

### A.1.4 Ouverture et fermeture

**Définition 42** *L'ouverture par l'élément structurant $B$ est constitué d'une érosion par $B$ suivi d'un dilatation par $-B$ :*

$$X_B = D(E(X, B), -B) = (X \ominus (-B)) \oplus B$$

**Propriété 8** *L'ouverture supprime les parties plus petites que l'élément structurant et lisse les contours. Elle est anti-extensive :*

$$X_B \subset X$$

*Croissante :*

$$X \subset Y \Rightarrow X_B \subset Y_B$$

*Idempotente :*

$$(X_B)_B = X_B$$

**Définition 43** *La fermeture par l'élément structurant $B$ est constitué de la dilatation par $B$ suivi de l'érosion par $-B$ :*

$$X^B = E(D(X, B), -B) = (X \oplus (-B)) \ominus B$$

**Propriété 9** *La fermeture bouche les trous plus petits que l'élément structurant et lisse les contours. Elle est extensive :*

$$X_B \subset X \subset X^B$$

*Croissante :*

$$X \subset Y \Rightarrow X^B \subset Y^B$$

*Idempotente :*

$$(X^B)^B) = X^B$$

*et duale de l'ouverture :*

$$\neg(X^B) = (\neg X)^B$$

### A.1.5 Transformation en tout-ou-rien

La notion de **transformée en tout-ou-rien** [1] unifie et généralise les opérateurs d'érosion et de dilatation. Ici l'élément structurant est un couple $B = (H, M)$ de parties de E tels que $H \cap M = \emptyset$.

**Définition 44** *La transformée en tout ou rien de A par le couple $B = (H, M)$ est l'ensemble :*

$$\begin{aligned} A \otimes B &= \{x \,|H_x \subset A \, et \, A \cap M_x = \emptyset\} \\ &= \{x \,|H_x \subset A \, et \, M_x \subset \neg A\} \\ &= E(A, H) \cap E(\neg A, M) \\ &= (A \ominus \breve{H}) \cap (\neg A \ominus \breve{M}) \end{aligned}$$

La TTR correspond à la recherche de configurations particulières dans une image binaire : l'ensemble $H$ représente les points à 1 d'une configuration et l'ensemble $M$ les points à 0, et l'origine prend la valeur 1 par la TTR. La TTR est équivalente à un monôme conjonctif[2] dont les variables sont les éléments de $H_x$ et les éléments de $M_x$ complémentés. Or, tout opérateur booléen local[3] commutant avec les translations, peut s'écrire sous forme normale disjonctive, c'est-à-dire se décomposer en disjonction de monômes conjonctifs. Donc tout opérateur booléen invariant en translation peut se décomposer en un ensemble de transformée en tout-ou-rien.

**Propriété 10** *Si $M = \emptyset$, nous avons $B = (H, \emptyset)$ et nous retrouvons l'érosion d'élément structurant $H$ :*

$$A \otimes B = \{x \,|H_x \subset A\} = E(A, B) = A \ominus \breve{H}$$

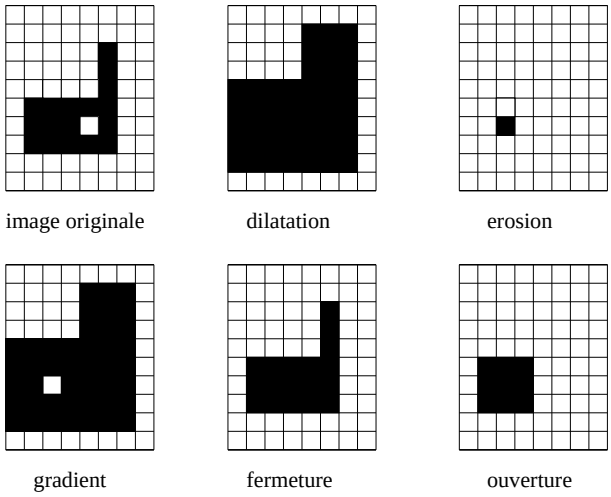



FIG. A.2 – Opérations élémentaires de morphologie mathématique binaire en 8-connexité pour un élément structurant de rayon 1.

### A.1.6 Amincissement et épaississement

**Définition 45** *Étant donné l'élément structurant $B = (H, M)$, l'amincissement est définie comme :*

$$X \circ B = X - (X \otimes B)$$

*et l'épaississement comme :*

$$X \odot S = X \cup (X \otimes S)$$

**Propriété 11** *L'amincissement est anti-extensif et l'épaississement est extensif. L'amincissement supprime de $X$ les points sélectionnés par la TTR et l'épaississement les rajoute.*

La Figure A.2 illustre les principales opérations élémentaires de la morphologie mathématique binaire.

[1]TTR ou "hit or miss" en anglais
[2]opérateur booléen faisant intervenir uniquement l'opérateur de conjonction
[3]le résultat ne dépend que des valeurs sur un voisinage de taille borné du pixel courant

## A.2 Morphologie en niveaux de gris

Les ensembles de niveaux représentent une manière naturelle d'étendre les opérateurs morphologiques binaires aux images en niveaux de gris. Une image binaire étant définie comme un sous-ensemble de $\mathbb{Z}^2$, on définit une image numérique en $n$ niveaux de gris par une fonction de $\mathbb{Z}^2$ dans $\{0, ..., n-1\}$. Une telle image en niveaux de gris $I$ peut-être représentée par ses ensemble de niveaux $\{I_t\}_{t\in[1,n]}$ définis ainsi :

**Définition 46 (image en niveaux de gris)**

$$I \equiv \{I_1, ..., I_{n-1}\}$$

*avec* $I_t = \{p \in E | I(p) \geq t\}$*, et réciproquement :*

$$I(p) = \max\{t; p \in I_t\}$$

**Propriété 12** *Le prolongement d'un opérateur morphologique* $\Omega$ *défini dans* $\{0, 1\}$ *à l'opérateur* $\breve{\Omega}$ *défini dans* $\{0, ..., n-1\}$ *est simplement donné par :*
*Si* $I \equiv \{I_1, ..., I_{n-1}\}$ *alors* $\breve{\Omega}(I) \equiv \{\Omega(I_1), ..., \Omega(I_{n-1})\}$.

Le passage direct d'un opérateur binaire vers un opérateur en niveaux de gris n'est possible que si l'opérateur est croissant. En fait, tous les opérateurs morphologiques peuvent se décomposer en un petit nombre d'opérations booléennes effectuées sur les seuillages successifs de l'image acquise.

### A.2.1 Dilatation et érosion

**Définition 47** *L'image* $D(X, B)$ *dilatée de l'image numérique* $X$ *par l'élément structurant* $B \subset \mathbb{Z}^2$ *est définie, pour tout* $z \in \mathbb{Z}^2$ *:*

$$D(X, B)(z) = (X \oplus \breve{B})(z) = \sup_{b\in B}\{X(z - b)\}$$

*L'image* $E(f, B)$ *érodée de l'image* $X$ *par l'élément structurant* $B \subset \mathbb{Z}^2$ *est définie, pour tout* $z \in \mathbb{Z}^2$ *:*

$$E(X, B)(z) = (X \ominus \breve{B})(z) = \inf_{b\in B}\{X(z - b)\}$$

### A.2.2 Ouverture et fermeture

**Définition 48** *Les opérateurs d'ouverture et de fermeture sont définies respectivement par :*
$\gamma_B(X) = (X \ominus \breve{B}) \oplus B$ *et* $\phi_B(X) = (X \oplus \breve{B}) \ominus B$

**Propriété 13** *Ces opérateurs sont croissants et idempotents. De plus, si $B$ contient l'origine alors* $X \ominus \breve{B} \leq \gamma_B(X) \leq X \leq \phi_B(X) \leq X \oplus \breve{B}$

### A.2.3 Gradient morphologique

On définit aussi le gradient morphologique qui est calculable en effectuant la différence (ensembliste) sur chaque seuil entre le dilaté et l'érodée :

**Définition 49 (gradient morphologique)**

$$g_B(X) = (X \oplus \breve{B}) - (X \ominus \breve{B})$$

On peut décomposer le gradient en la somme du gradient interne $g^i_B(X) = X - (X \ominus \breve{B})$ et du gradient externe $g^e_B(X) = (X \oplus \breve{B}) - X$.

### A.2.4 Chapeau haut-de-forme

La transformation en chapeau haut-de-forme de l'image $X$ par l'élément structurant binaire $B$ est définie par $X - \gamma_B(X)$. Cet opérateur détecte les parties fines et claires de l'image. Le haut-de-forme conjugué est définie par $\phi_B(X) - X$.

La Figure A.3 présente un exemple des principales fonctions morphologiques en niveaux de gris décrites dans cette section.

### A.2.5 Reconstruction géodésique

L'opérateur que nous décrivons ici a des propriétés intéressantes puisqu'il n'introduit pas de nouveaux contours de l'image, contrairement aux ouvertures/fermetures morpho-

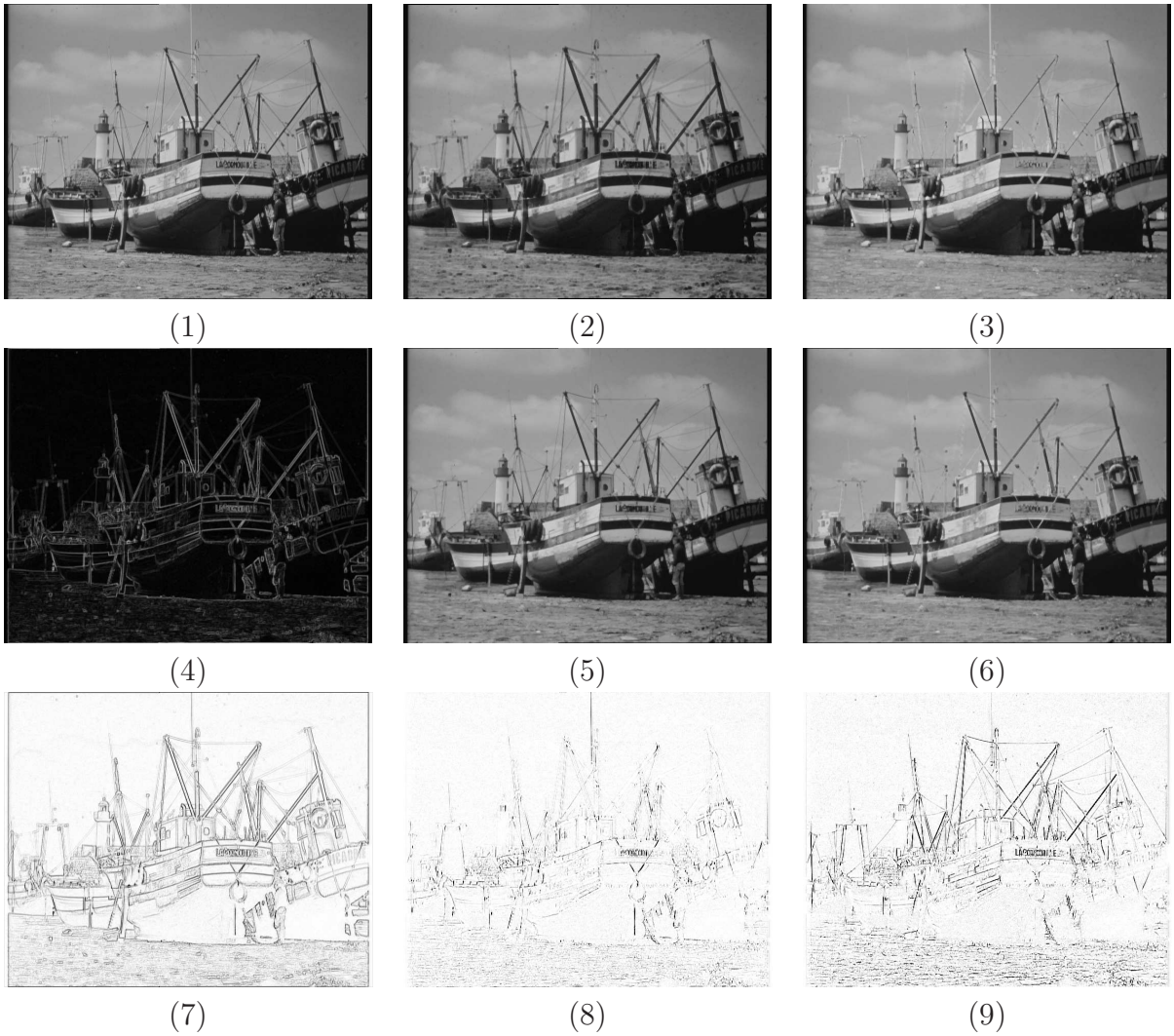


FIG. A.3 – Les opérateurs morphologiques de base en niveaux de gris. Pour toutes ces opérations, l'élément structurant est le voisinage 3x3 autour de l'origine. (1) Image originale (boats) (2) Érosion (3) Dilatation (4) Gradient morphologique (5) Ouverture (6) Fermeture (7) Amincissement (présenté en mode vidéo inversée) (8) Chapeau haut-de-forme (présenté en mode vidéo inversée) (9) Haut-de-forme conjugué (présenté en mode vidéo inversée)

logiques. Soient $f_{inf}$ et $f_{sup}$ deux images telles que $f_{inf} \leq f_{sup}$ , c.à.d $\forall p \in E; f_{inf}(p) \leq f_{sup}(p)$.

**Définition 50** *La reconstruction géodésique de $f_{sup}$ par dilatation de $f_{inf}$, notée $rec_{\oplus}(f_{inf}, f_{sup})$ est la limite de la suite convergente :*

$$\begin{cases} f_0 = f_{inf} \\ f_{n+1} = (f_n \oplus B_u) \wedge f_{sup} \end{cases}$$

*où $B_u$ est la boule unité définie dans $E$.*

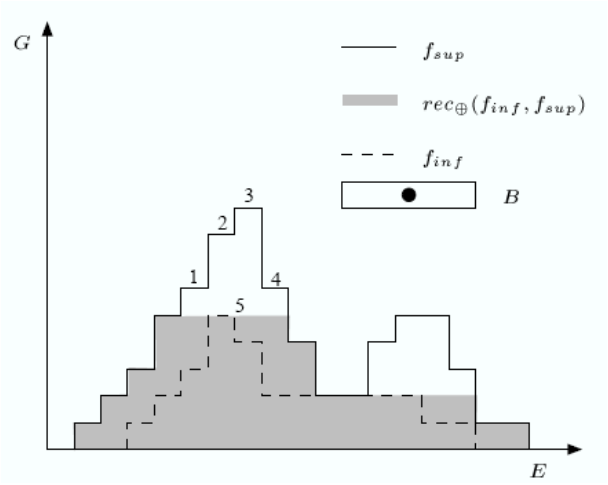



FIG. A.4 – Exemple de reconstruction géodésique de $f_{sup}$ par dilatation de $f_{inf}$.

**Définition 51** *Par dualité on définit la reconstruction géodésique de $f_{inf}$ par érosion de $f_{sup}$ : $rec_{\ominus}(f_{sup}, f_{inf}) = rec_{\oplus}(f_{sup}^c, f_{inf}^c)^c$. C'est l'image limite de la suite convergente :* $\begin{cases} f_0 = f_{sup} \\ f_{n+1} = (f_n \ominus B_u) \vee f_{inf} \end{cases}$ *en effet $\check{B}_u = B_u$.*

Un exemple est présenté sur la Figure A.4 : on dilate $f_{inf}$ au fur et à mesure en restant sous l'enveloppe de $f_{sup}$ à chaque itération, jusqu'à convergence. Une des propriétés fondamentales de la reconstruction géodésique est de simplifier l'image en fusionnant les zones plates (régions où le niveau de gris est constant). Sur la Figure A.4 les zones plates numérotées de 1 à 4 ont fusionné pour donner la zone plate 5 sur l'image après reconstruction. La conséquence est qu'elle préserve les contours des objets de l'image. Lorsque

$f_{inf}$ (respectivement $f_{sup}$) est obtenue par érosion (resp. dilatation) de $f_{sup}$ (resp. $f_{inf}$) par un élément structurant plat connexe alors $rec_{\oplus}(f_{inf}, f_{sup})$ est appelée une ouverture géodésique (resp. $rec_{\ominus}(f_{sup}, f_{inf})$ une fermeture géodésique) [ARH00][Sal02] [Vin93].

**Propriété 14** *Par construction nous avons* $f_{inf} \leq rec_{\ominus}(f_{sup}, f_{inf}) \leq f_{sup}$ *et* $f_{inf} \leq rec_{\oplus}(f_{inf}, f_{sup}) \leq f_{sup}$.

FIG. A.5 – La reconstruction géodésique est définie par l'ensemble des composantes connexes de l'image qui intersectent le marqueur matérialisé par le rectangle bleu.

Dans le cadre ensembliste, la dilatation géodésique de $X$ par $R$ (par la boule élémentaire) devient l'ensemble des voisins de $X$ inclus dans $R$ :

$$\delta_B^R(X) = \delta_B(X) \cap R$$

Et la reconstruction géodésique ensembliste de $X$ dans $R$ est l'ensemble des composantes connexes de $R$ voisines de $X$.

La Figure A.5 illustre la reconstruction géodésique dans le cadre ensembliste.

# Annexe B

# Les squelettes en niveau de gris

Parmi les travaux majeurs concernant le calcul d'un squelette directement en niveaux de gris nous pouvons citer [Dok00] [Bez01] [CBB01]. Ces travaux sont basés sur des notions topologiques dont le noyau homotopique. Les opérations de squelettisation et d'amincissement sont en effet les applications majeures des calculs topologiques en niveaux de gris.

**Définition 52 (Relation de voisinage)** *Nous considérons les deux relations de voisinage* $\Gamma_4$ *et* $\Gamma_8$ *définie par, pour chaque point* $x \in \mathbb{Z}^2$ *:*
$\Gamma_4(x) = \{y \in \mathbb{Z}^2; |y_1 - x_1| + |y_2 - x_2| \leq 1\}$ *et* $\Gamma_8(x) = \{y \in \mathbb{Z}^2; max(|y_1 - x_1|, |y_2 - x_2|) \leq 1\}$.
*Dans la suite, nous utilisons la notion de voisinage* $\Gamma_n^*(x) = \Gamma_n(x) \backslash \{x\}$*, avec* $n = 4$ *ou* $8$.

Le point $y \in \mathbb{Z}^2$ est $n$-adjacent de $x \in \mathbb{Z}^2$ si $y \in \Gamma_n^*(x)$.
un $n$-chemin est une séquence de points $x_0, \ldots, x_k$ où $x_i$ est $n$-adjacent de $x_{i-1}$ pour $i = 1, \ldots, k$.

## B.1 Topologie pour les images en niveau de gris

Soit $X \subset \mathbb{Z}^2$, $x \in \mathbb{Z}^2$, nous notons $\bar{X}$ le l'ensemble complémentaire de $X$. Soit $C_n[x, X]$ l'ensemble composé de toutes les composantes $n$-connectées de $X$ $n$-adjacente à $x$. Nous posons que deux points $x$ et $y$ de $\mathbb{Z}^2$ sont $n$-connectés dans $X$ s'il existe un $n$-chemin

dans $X$ entre ces 2 points. Ceci défini une relation d'équivalence. $y$ est n-adjacent de $x$ si $y \in \Gamma_n^*(x)$.

Afin d'avoir une correspondance entre la topologie de $X$ et celle de $\bar{X}$ et de rester en accord avec le Théorème de Jordan en discret, nous devons considérer deux types différents d'adjacences pour $X$ et $\bar{X}$. Ainsi si nous utilisons une $n$-adjacence pour $X$, nous devons utiliser une $\bar{n}$-adjacence pour $\bar{X}$, avec $(n, \bar{n}) = (4, 8)$ ou $(8, 4)$. Nous considérons dans les algorithmes suivant que les configurations d'adjacence sont correctement choisies afin de respecter cette contrainte.

Soit $T$ le nombre de topologie, défini par $T(x, X) = \sharp C_n[x, \Gamma_8^*(x) \cap X]$, le nombre de composantes 8-connexes dans $X$. Ainsi $\bar{T}(x, X) = \sharp C_{\bar{n}}[x, \Gamma_8^*(x) \cap \bar{X}]$ est le nombre de composantes 8-connexes dans $\bar{X}$. Si $T(x, X) = 0$, il n'y a aucune composante connexe dans $X$, $x$ est donc un point isolé dans $X$. Si $\bar{T}(x, X) = 0$, il n'y a aucune composante connexe dans $\bar{X}$, $x$ est un point intérieur de $\bar{X}$. Si $\bar{T}(x, X) > 0$, il existe au moins une composante connexe dans $\bar{X}$, $x$ est un point de bordure. Enfin, la propriété 15 nous permet de caractériser localement les points simples.

**Définition 53 (Point simple)** *Soit $x \in X$, $K = 4$ ou 8, $x$ est un point $K$-simple de $X$ si et seulement si :*

- *$x$ a au moins un $\tilde{K}$-voisin dans $X^c$ ;*
- *$x$ est $K$-voisin d'une seule $K$-composante connexe dans son voisinage strict.*

*$x$ est $K$-simple dans $X \Leftrightarrow N_{c_k}^X(x) = 1$ (voir définition 12).*

**Propriété 15 (Caractérisation locale des points simples)**

$$x \in \mathbb{Z}^2 \text{ est simple pour } X \subset \mathbb{Z}^2 \Leftrightarrow T(x, X) = 1 \text{ et } \bar{T}(x, X) = 1$$

Nous rappelons qu'un point $x \in X$ est simple si la suppression de ce point ne modifie pas la topologie de l'ensemble $X$. D'autres configuration sont rassemblées dans la Table B.1.

Soit $F \in \mathcal{F}$ et $k \in \mathbb{Z}$, la section de $F$ au niveau $k$ est l'ensemble $F_k$ composé de tout les points $x \in \mathbb{Z}^2$ tels que $F(x) \geq k$. Le point $x \in \mathbb{Z}^2$ est destructible si $x$ est simple pour $F_k$, $x$ est constructible si $x$ est destructible pour $-F$, le complémentaire de $F$[1].

[1] pour chaque point $x$ dans $\mathbb{Z}^2$, $(-F)(x) = -F(x)$

| | | |
|---|---|---|
| $T = 0$ | - | point isolé |
| - | $\bar{T} = 0$ | point intérieur |
| $T = 1$ | $\bar{T} = 1$ | simple point de bordure |
| $T = 2$ | $\bar{T} = 1$ | isthme 1-D |
| $T = 1$ | $\bar{T} = 2$ | isthme 2-D |
| $T \geq 3$ | $\bar{T} = 1$ | jonction d'isthme 1-D |
| $T = 1$ | $\bar{T} \geq 3$ | jonction d'isthme 2-D |
| $T \geq 2$ | $\bar{T} \geq 2$ | jonction d'isthme hybride |

TAB. B.1 – Classification topologique des points.

Nous observons qu'une section est un ensemble de points, c'est-à-dire une image binaire. Comme cela était déjà le cas en binaire, si nous utilisons la $n$-adjacence pour les sections $F_k$ de $F$, nous devons utiliser la $\bar{n}$-adjacence pour les ensembles complémentaires $\overline{F_k}$.

Intuitivement, une transformation sur $F$ conserve la topologie si la topologie de toutes les sections de $F$ sont préservées. Ainsi, nous concluons que la valeur du niveau de gris d'un point destructible (resp. constructible) peut être abaissé (resp. remonter) d'une unité tout en conservant la topologie de $F$.

## B.2 Caractérisations locales

Soit $F \in \mathcal{F}$ et $x \in \mathbb{Z}^2$, nous définissons les quatre voisinages :
$\Gamma^{++}(x, F) = \{y \in \Gamma_8^*(x);\ F(y) > F(x)\}$ ;
$\Gamma^{--}(x, F) = \{y \in \Gamma_8^*(x);\ F(y) < F(x)\}$ ;
$\Gamma^{+}(x, F) = \{y \in \Gamma_8^*(x);\ F(y) \geq F(x)\}$ ;
$\Gamma^{-}(x, F) = \{y \in \Gamma_8^*(x);\ F(y) \leq F(x)\}$.

Nous définissons :
$\beta^+(x, F) = max\{F(y);\ y \in \Gamma(x)\}$ ;
$\beta^-(x, F) = min\{F(y);\ y \in \Gamma(x)\}$, et

$$\alpha^+(x, F) = \begin{cases} min\{F(y);\ y \in \Gamma^{++}(x, F)\} \text{ si } \Gamma^{++}(x, F) \neq \emptyset \\ F(x) \text{ sinon} \end{cases} ;$$

$$\alpha^-(x, F) = \begin{cases} max\{F(y);\ y \in \Gamma^{--}(x, F)\} \text{ si } \Gamma^{--}(x, F) \neq \emptyset \\ F(x) \text{ sinon} \end{cases} .$$

Pour un point $x$, nous définissons la valeur $\delta^-(x, F)$ la valeur minimale à laquelle un

point $x$ peut être abaissé sans changer la topologie des sections. Ainsi, pour un point destructible, nous avons $\delta^-(x,F) < F(x)$, et pour un point non-destructible, nous avons $\delta^-(x,F) = F(x)$. La valeur $\delta^+(x,F)$ étant définie de manière duale.

Nous définissons les quatre nombres de connexité :
$T^{++}(x,F) = \sharp C_n[x, \Gamma^{++}(x,F)] = T^{++}$ ;
$T^{--}(x,F) = \sharp C_n[x, \Gamma^{--}(x,F)] = T^{--}$ ;
$T^{+}(x,F) = \sharp C_n[x, \Gamma^{+}(x,F)] = T^{+}$ ;
$T^{-}(x,F) = \sharp C_n[x, \Gamma^{-}(x,F)] = T^{-}$.

Les propriétés 16 sont directement dérivées des définitions précédentes et de la caractérisation des points simples dans des images binaires. Le nombre de connexité permet de caractériser localement les points destructibles et constructibles.

**Propriété 16** *Soit $F \in \mathcal{F}$ et $x \in \mathbb{Z}^2$*
*$x$ est destructible pour $F \Longleftrightarrow T^+ = 1$ et $T^{--} = 1$ ;*
*$x$ est constructible pour $F \Longleftrightarrow T^- = 1$ et $T^{++} = 1$.*
*$x$ est un pic si $T^+ = 0$ ; $x$ est minimal si $T^{--} = 0$ ; $x$ est $k$-divergent si $T^{--} = k$ ;*
*$x$ est un creux si $T^- = 0$ ; $x$ est maximal si $T^{++} = 0$ ; $x$ est $k$-convergent si $T^{++} = k$.*
*$x$ est un point inférieur si il n'est pas maximal, supérieur s'il n'est pas minimal.*
*$x$ est un point intérieur si il est à la fois maximal et minimal.*
*$x$ est simple si il est à la fois destructible et constructible.*
*$x$ est un point de jonction ("saddle") si il est à la fois convergent et divergent.*

En considérant toutes les valeurs possibles des nombres en 4-connexité, nous pouvons voir que pour $F \in \mathcal{F}$, un point $x \in \mathbb{Z}^2$ correspond nécessairement à une et seulement une des configurations suivantes :

1. un pic ;
2. un point isolé ;
3. un point intérieur ;
4. un point minimal constructible ;
5. un point maximal destructible ;
6. un point minimal convergent ;
7. un point minimal convergent ;
8. un point maximal divergent ;
9. un point simple ;
10. un point convergent destructible ;

11. un point divergent constructible ;
12. un point *saddle* ;

La figure B.1 nous montre certaines de ces configurations.

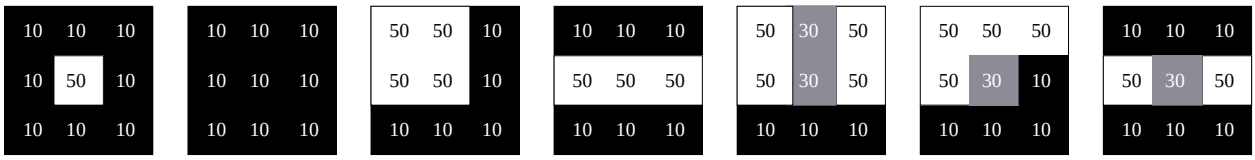



FIG. B.1 – Type topologique (a) un pic ; (b) un point intérieur ; (c) un point maximal destructible ; (d) un point maximal 2-divergent ; (e) un point destructible 2-convergent ; (f) un point simple côté ; (g) un point *saddle* .

## B.3 Notion de point λ-destructible

Nous introduisons la notion de point λ-destructible qui nous permet d'altérer de manière sélective la topologie et donc d'éliminer (d'abaisser le niveau de gris) un nombre plus important de points afin d'avoir une meilleure approximation du squelette en niveau de gris. Le paramètre utilisé, $\lambda$, représente une mesure locale du contraste.

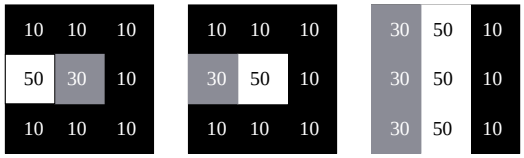
**Définition 54 (Point λ-destructible)** *Soit $F \in \mathcal{F}, x \in \mathbb{Z}^2$, nous définissons $F^-(X) = min\{F(x), x \in X\}$. Le point $x$ est $\lambda$-destructible, $\lambda \in \mathbb{N}^{+*}$, si il satisfait les deux contraintes suivantes :*

1. *$x$ est destructible ou*
2. *$x$ est $k$-divergent et au moins $k-1$ composantes connexes $c_i, i = 1, \ldots, k-1$ de $\Gamma^{--}(x, F)$ sont telles que $F(x) - F^-(C_i) \leq \lambda$*

Intuitivement, la seconde contrainte de la Définition 54 établit que le point $x$ appartient à une crête qui sépare son voisinage en $k$ régions distinctes, et au moins une de ces régions est à une différence de niveaux de gris par rapport à $x$ supérieure à $\lambda$. La notion de point $\lambda$-constructible peut-être définie de manière duale à partir de la Définition 54.

## B.4 Squelette en niveaux de gris

Soit $X \subset \mathbb{Z}^2$ et $x \in X$, $x$ est une extrémité pour $X$ si $\sharp(\Gamma_n^*(x) \cap X) = 1$. Soit $F \in \mathcal{F}$ et $x \in \mathbb{Z}^2$, $x$ est une extrémité pour $F$ si il est une extrémité pour l'ensemble $F_k$ avec $k = F(x)$. Un point est dit $\lambda$-final pour $F$ si il est une extrémité pour $F$ et si $F(x) - \lambda^-(x, F) > \lambda$. Un point est dit un point $\lambda$-supprimable pour $F$ si il est à la fois $\lambda$-destructible ou un pic tel que $F(x) - \lambda^-(x, F) \leq \lambda$.




(a) (b) (c)

FIG. B.2 – Exemple de points $\lambda$-final et $\lambda$-supprimable. (a) : le point central est un point 29-final ; (b) et (c) : le point central est un point 20-supprimable .

Si $X \subset \mathbb{Z}^2$ représente une image binaire, nous pouvons dire que $S \subset \mathbb{Z}^2$ est un squelette de $X$ si $S$ peut être obtenue depuis $X$ en sélectionnant itérativement un point simple non final de $X$ et en le supprimant jusqu'à stabilité. Ainsi de manière similaire, pour $F \in \mathcal{F}$, $G \in \mathcal{F}$ est une squelette de $F$ si $G$ peut être obtenu depuis $F$ en sélectionnant itérativement un point destructible non final de $F$ et en abaissant son niveau de gris à $\alpha^-(x, F)$ jusqu'à stabilité. Afin d'obtenir un squelette filtré, c'est-à-dire d'éliminer toutes les branches non significatives et les minima régionales, nous devons permettre l'abaissement de points $\lambda$-supprimables non $\lambda$-finaux.

La Table B.2 nous montre l'algorithme utilisé afin d'abaisser les points $\lambda$-destructibles permettant de filtrer le squelette en niveau de gris.

**entrée** : $F \in \mathcal{F}$ et $\lambda \in \mathbb{N}$ **sortie** : $F$
**Répéter** jusqu'à stabilité **{**
  **Pour** chaque point $\lambda$-destructible ou pic de $F$ **{**
    Choisir un point $x$ de valeur minimal
      $F(x) := \alpha^-(x, F)$
  **}**
**}**

TAB. B.2 – Algorithme d'abaissement des points $\lambda$-destructibles.

Finalement, la Table B.3 présente l'algorithme de calcul du $\lambda$-squelette en niveau de gris.

**entrée** : $F \in \mathcal{F}$ et $\lambda \in \mathbb{N}$ **sortie** : $F$
**Répéter** jusqu'à stabilité **{**
  **Pour** chaque point $\lambda$-supprimable et non $\lambda$-final de $F$ **{**
    Choisir un point $x$ de valeur minimal
      $F(x) := \alpha^{-}(x, F)$
    **}**
  **}**

TAB. B.3 – Algorithme de calcul du $\lambda$-squelette en niveaux de gris.

# Bibliographie


[AB86] H. Asada and M. Brady. The curvature primal sketch. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 8(1) :2–14, January 1986.

[AB96] A.G. Andreou and K.A. Boahen. Translinear circuits in subthreshold cmos. 9(2) :141–166, March 1996.

[ABC98] G. Adorni, F. Bergentil, and S. Cagnoni. Vehicule license plate recognition by means of cellular automate. *IEEE*, pages 689–693, 1998.

[AD91] N. Ansari and E.J. Delp. On detecting dominant points. *Pattern Recognition*, 24(5) :441–451, 1991.

[AG92] A. Arnold and I. Guessarian. *Mathématiques pour l'informatique*. Masson, 1992.

[Agn04] V. Agnus. *Segmentation spatio-temporelle de séquences d'images par des opérateurs de morphologie mathématique*. PhD thesis, Université de Rennes 1, Novembre 2004.

[AM] P.C. Arribas and F. Monasterio-Huelin Macia. Fpga implementation of the horn & shunck optical flow algorithm for motion detection in real time images.

[Anc02] F. Anceau. L'évolution de l'architecture des microprocesseurs. In *Colloque international : Techniques et technologies fondamentales de la nouvelle conomie*, Albena, Bulgary, September 2002.

[APS02] P. Agouris, P. Parstinevelos, and A. Stefanidis. Differentiating and modeling multiple moving objects in motion imagery datasets. *Interntional Archives of the Photometry, Remote Sensing and Spatial Information Sciences*, 34, 2002.

[ARH00] V. Agnus, C. Ronse, and F. Heitz. Spatio-temporal segmentation using morphological tools. In *15th ICPR*, pages 885–888, Barcelona, Spain, September 2000.

[Arn01] E. Arnaud. *Méthodes de filtrage pour du suivi dans les séquences d'images. Application au suivi de points caractéristiques.* PhD thesis, LSIIT, Université Louis Pasteur, Strasbourg, Octobre 2001.

[AS97] A. Amin and S. Singh. Machine recognition of hand-printed chinese characters. 1 :101–118, 1997.

[Aub04] O. Aubreton. *Rétines à masques : utilisation de masques binaires pour l'implantation d'un opérateur de reconnaissance de forme dans une rétine CMOS.* PhD thesis, Université de Bourgogne, Décembre 2004.

[BA92] K.A. Boahen and A.G. Andreou. A contrast sensitive silicon retina with reciprocal synapsis. *Neural Information Processing Systems*, 1992.

[BB82] D.H. Ballard and C.M. Brown. *Computer vision.* Prenctice-Hall, 1982.

[BB92] A. Belaid and Y. Belaid. *Reconnaissance des formes.* InterEditions, 1992.

[BB95] S.S. Beauchemin and J.L. Barron. The computation of optical flow. In *ACM Computing Surveys*, volume 27(3), pages 434–467, September 1995.

[Bea78] P.R. Beaudet. Rotationnaly invariant image operators. In *Proc. of the 4th IJCPR*, pages 579–583, Tokyo, Japan, 1978.

[Bel96] A. Bellon. *Détection et suivi de véhicules en mouvement dans une séquence d'images.* PhD thesis, Université Blaise Pascal, Clermond-Ferrand, Octobre 1996.

[Ber92] T.M. Bernard. *Des Rétines artificielles intelligentes.* PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, Octobre 1992.

[Ber97] T.M. Bernard. Rétines artificielles : Quelle intelligence au sein du pixel ? In B. Pottier et al., editors, *Calculateurs Parallèlles, special issue on FPGAs and Smart Sensors*, pages 77–108. Hermès, Paris, march 1997.

[Ber98] T.M. Bernard. Les rétines artificielles programmables prêtes à sortir du laboratoire. *Revue de l'Electricité et de l'Electronique*, (10) :93–99, november 1998.

[Bez01] F. Bezerra. *Opérateurs topologiques pour le traitement d'images en niveau de gris.* PhD thesis, université Marne-la-Vallée, Novembre 2001.

[BJ98] S. Bres and J-M. Jolion. Multiresolution contrast based detection of interest points. Technical report, INSA, 1998.

[BJ99] S. Bres and J-M. Jolion. Detection of interest points for image indexation. In *3rd Int. Conf. on Visual Information Systems*, Amstedam, Holland, June 1999.

[BKW98] Backhaus, Kliegl, and Werner. *Color vision, perspectives from different disciplines.* De Gruyter, 1998.

[BL93] P. Bouthemy and P. Lalande. Recovery of moving objects masks in an image sequence using local spatiotemporal contextual information. *Optical Engineering*, 32(6) :1205–1212, June 1993.

[Blu67] H. Blum. *A transformation for extracting new descriptors of shape, Symp. Models for perception of speech and visual form*. Weiant Whaten-Dunn Ed., MIT press, Cambrodge, 1967.

[BM99] T.M. Bernard and A. Manzanera. Improved low complexity fully parallel thinning algorithm. In *Proc. Int. Conf. on Image Analysis and Processing*, pages 215–220, Venice, Italy, september 1999. IEEE Computer Society.

[Bon91] P. Bonnin. *Méthode systématique de conception et de réalisation d'applications en vision par ordinateur*. PhD thesis, Université Paris 7, 1991.

[BP97] T.M. Bernard and F. Paillet. Output methods for an associative operation of programmable artificial retina. In *IEEE/RJS Int. Conf. on Intelligent Robots and Systems*, pages 752–757, September 1997.

[BP00] E. Bruno and D. Pellerin. *Robust motion detection using spatial Gabor filters*. EUS, 2000.

[BREJC97] J. Barroso, A. Rafael, E.L.Dagless, and J.Bulas-Cruz. Number plate reading using computer vision. *IEEE*, pages 761–766, 1997.

[Bru01] E. Bruno. *De l'estimation locale à l'estimation globale de mouvement dans les séquences d'images*. PhD thesis, Université Joseph Fourier, Grenoble, Novembre 2001.

[BSNS98] M.H. Brugge, J.H. Stevens, J.A.G. Nijhuis, and L. Spaanenburg. License plate recognition using dtcnns. *IEEE*, pages 212–217, 1998.

[BT00] R. Beare and H. Talbot. Motion segmentation using seeded region growing. In *International Symposium on Mathematical Morphology*, June 2000.

[BW05] A. Bruhn and J. Weickert. Lucas/kanade meets horn/schunck : Combining local and global optic flow methods. *International Journal of Computer Vision*, 6(13) :211–231, 2005.

[Can86] J.F. Canny. A computionnal approach to edge detection. 8(6) :679–698, 1986.

[CBB01] M. Couprie, F.N. Bezerra, and G. Bertrand. Topological operators for grayscale image processing. 2001.

[CD] M.L. Comer and E.J. Delp. Morphological operations for color image processing.

[CDLC96] A. Caplier, C. Dumontier, F. Luthon, and P-Y. Coulon. Algorithme de détection de mouvement par modélisation markovienne. mise en œuvre sur dsp. 13(2) :178–190, 1996.

[CGE+96] H.H. Cat, A. Gentille, J.C. Eble, M. Lee, O. Vendier, Y.J. Doo, D. Scott Wills, M. Brooke, N.M. Jokerst, and A.S. Brown. Simpil : An integrated simd architecture for focal plane processing applications. In *Proc. of Third Int. Conf. on Massively Parallel Processing using Optical Interconnections*, pages 44–52, Maui, Hawaii, USA, October 1996.

[CGP03] R. Cucchiara, C. Grana, and A. Prati. Effective detection of moving objects, shadows and ghosts in surveillance videos. In *Australia-Japan Advanced Worshop on Compuiter Vision*, December 2003.

[CK03] S-C.S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Nice, France, 2003.

[CKHL03] T.H. Chalidabhongse, K. Kim, D. Harwood, and L.Davis. A perturbation method for evaluating background substraction algorithms. Nice, France, 2003.

[CLK99] R.T. Collins, A.J. Lipton, and T. Kanade. A system for video surveillance and monitoring. 1999.

[CP99] V. Chatzis and I. Pitas. Shape-based interpolation of binary 3-d images using morphological skeletonization. In *IEEE Int. Conf. on Multimedia Computing and Systems*, volume II-2, Florence, Italy, 1999.

[CR93] L.O. Chua and T. Roska. The cnn paradigm. 40(3) :147–156, 1993.

[CS00] L. Czùni and T. Szirànyi. Motion segmentation and tracking optimization with edge relaxation in the cellular nonlinear network architecture. In *Proc. of the 6th IEEE Int. Work. on CNNA*, pages 51–56, Catenia, Italy, May 2000.

[CT97] R. Cardoner and F. Thomas. Residuals + directional gaps = skeletons. 18 :343–353, 1997.

[CVK93] J. Cooper, S. Venkatesh, and L. Kitchen. Early jump-out corner detectors. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 15(8) :823–833, 1993.

[CY89] M-H. Chen and P-F. Yan. A multiscale appoach based on morphological filtering. 11 :694–700, 1989.

[Del83] T. Delbrück. Silicon retina with correlation-based, velocity-tuned pixels. *IEEE Trans. on Neural Networks*, 4(3) :529–541, May 1983.

[DeW92] S.P. DeWeerth. Analog vlsi circuits for stimulus localization and centroid computation. 8(3) :191–202, September 1992.

[DG93] R. Deriche and G. Giraudon. A computational approach for corner vertex detection. 10(2) :101–124, 1993.

[DG04] R. Dinesh and D. Guru. Mathematical morphology based corner detection scheme : a non-parametric approach. Kolkata, India, December 2004.

[DGZ85] F. Devos, P. Garda, and B. Zavidovique. Rétine integrée à réseau de processeurs. In *Brevet N4-85-09256*. Juin 1985.

[DH96] K.M. Dawson-Howe. Active surveillance using dynamic background subtraction. August 1996.

[DH00] P. Dudek and J. Hicks. A cmos general-purpose sampled-data analogue microprocessor. Genève, Swiss, May 2000.

[DKHS98] K. Daniilidis, C. Krauss, M. Hansen, and G. Sommer. Real-time tracking of moving objects with an adaptative camera. *Real-Time Imaging*, 4 :3–20, 1998.

[DMLM05] J. Denoulet, J. Mostafaoui, L. Lacassagne, and A. Mérigot. Implementing motion markov detection on general purpose processor and associative mesh. 2005.

[DO96] A. Dupret and Others. An optoelectronic cmos circuit implementing a simulated annealing algorithm. 31(7) :1046–1050, july 1996.

[Dok00] P. Dokládal. *Greyscale image segmentation : a topological approach*. PhD thesis, Université marne la vallée, Janvier 2000.

[Duc99] B. Ducourthial. *Un modèle de programmation a parallélisme de données pour algorithmes et données irregulières à primitives de calcul asynchrones*. PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, Janvier 1999.

[Dul96] D. Dulac. *Contribution au parallélisme massif en analyse d'image : Une architecture SIMD fondée sur la reconfigurabilité et l'asynchronisme*. PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, Janvier 1996.

[Dum96] C. Dumonthier. *Etude et mise en œuvre temps réel d'un algorithme de détection du mouvement par approche markovienne*. PhD thesis, INPG, 1996.

[EHD00] A. Elgammal, D. Hardwood, and L. Davis. Non-parametric model for background subtraction. Dublin, Ireland, 2000.

[Elo05] A. Elouardi. *Evaluation des rétines électroniques pour une définition architecturale d'un système monopuce (SoC) dédié à la vision embarquée*. PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, Mai 2005.

[ESA96] J.E. Eklund, C. Svensson, and A. Astrom. Vlsi implementation of a focal plane image processor - a realization of the near-sensor image processing concept. 4(3) :322–335, September 1996.

[FG87] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. pages 281–305, Interlaken , Switzerland, 1987.

[FMS97] E. Decenciere Ferrandiere, S. Marshall, and J. Serra. Application of the morphological geodesic reconstruction to image sequence analysis. *IEE Proceedings - Vision, Image and Signal Processing*, 144(6) :339–344, December 1997.

[FO83] R. Forchheimer and A. Odmark. A single chip linear array processor. volume 397, pages 425–430, 1983.

[FYYN70] K. Fukushima, M. Yamaguchi, M. Ysuda, and S. Nagata. An electronic model of the retina. volume 12(58), pages 1950–1951, December 1970.

[Gal02] Patrick Gallinari. Apprentissage numerique et réseaux de neurones. DEA IARFA, 2002.

[Gar84] P. Garda. *Vers une architecture intégrée de rétines B -codées par processeurs cellulaires*. PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, Juillet 1984.

[Gar86] M. Gardner. *Knotted doughnuts and other mathematical entertainments*. W.H. Freeman, New York, 1986.

[GCBV93] B. Giai-Checa, P. Bouthemy, and T. Vieville. Detection d'objets en mouvement. Technical report, INRIA, June 1993.

[GCN+98] B. Galvin, B. Mc Cane, K. Novins, D. Mason, and S. Mills. Recovering motion fields : an evaluation of eight optical flow algorithms. In *British Machine Vision Conference*, volume 1, pages 195–204, Southampton, UK, September 1998.

[GH92] Z. Guo and R.W. Hall. Fast fully parrallel thinning algorithms. 55(3) :317–328, May 1992.

[Gie05] V. Gies. *Asynchronisme dans les rétines artificielles*. PhD thesis, Université Paris-Sud, UFR Scientifique d'Orsay, Décembre 2005.

[Gol89] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publising Company, Massachussets, 1989.

[GW92] R.C. Gonzalez and R.E. Woods. *Digital image processing*. 1992.

[Ha90] F. Heitger and al. Simulation of neural contour mechanism : from simple to end-stopped cells. *Vision Research*, 10(2) :963–981, 1990.

[Hee87] D.J. Heeger. Model for the extraction of image flow. . *Opt. Soc. Am.*, A(4) :1455–1471, 1987.

[Hen96] C. Hennebert. *Analyse d'une scène dynamique avec une caméra en mouvement : système de détection de cibles mobiles*. PhD thesis, Institut National de Physique, Grenoble, 1996.

[Hil84] E.C. Hildreth. *The measurement of visual motion*. MIT Press, Massachussets, Cambridge, 1984.

[Hil87] E.C. Hildreth. The analysis of visual motion : from computational theory to neural mechanisms. *Annual Review of Neuroscience*, (10) :477–533, 1987.

[HKLM88] J. Hutchinson, C. Koch, J. Luo, and C. Mead. Computing motion using analog and binary resistive networks. *IEEE Computer*, 21(3) :52–63, March 1988.

[HKSL90] J.G. Harris, C. Koch, E .Staats, and J. Luo. Analog hardware for detecting discontinuities in early visions. *Int. Journal of computer vision*, (4) :211–223, 1990.

[HLS02] D. Hall, B. Leibe, and B. Schiele. Saliency of interest points under scale changes. 2002.

[HS80] B.K.P. Horn and B. Schunck. Determining optical flow. Technical report, Massachussetts Institute of Technology, Artificial Intelligence Laboratory, April 1980.

[HS88] C. Harris and M. Stephens. A combined corner and vertex descriptor. 1988.

[HSZ87] R.M. Haralick, S.R. Sternberg, and X .Zhuang. Image analysis using mathematical morphology. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 9(4) :532–550, July 1987.

[HV90] R. Horaud and F. Veillon. Finding geometric and relational structures in an image. In *Proc. First European Conference on Computer Vision*, pages 374–384, Antibes, France, 1990.

[IDE97] R.A. Peters II, E.R. Dougherty, and J.T. Astola Eds. Mathematical morphology for angle-valued images. *Proc of the SPIE, Nonlinear Image Processing VIII*, 3026 :84–94, February 1997.

[II95] R.A. Peters II. A new algorithm for image noise reduction using mathematical morphology. *IEEE trans. on Image Processing*, 4(3) :554–568, May 1995.

[III99] A.O. Hero III. On the problem of granulometry for a degraded boolean image model. In *Proc. of ICIP'99*, pages 16–20, Japan, 1999.

[JBHE02] R. Hirata Jr, J. Barrera, R.F. Hashimoto, and D.O. Dantas G.H. Esteves. Segmentation of microarray images by mathematical morphology. *Real-Time Imaging*, 8 :495–505, 2002.

[JC90] B-K. Jang and R.T. Chin. Analysis of thinning algorithms using mathematical morphology. *IEEE trans. on Pattern Analysis and Machine intelligence*, 12(6) :541–551, June 1990.

[JFS95] C.E. Jacobs, A. Finkelstein, and D.H. Salesin. Fast multiresolution image querying. In *Proceedings of SIGGRAPH*, Los Angeles, California, USA, 1995.

[JN79] R. Jain and H.H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. In *IEEE trans. on PAMI*, volume 206(18), pages 206–214, 1979.

[KB90] K.P. Karmann and A. Von Brandt. *Time-varying image processing and moving object recognition.* Elsevier, 1990.

[KDMA99] J.O. Klein, A. Dupret, A. Moutault, and A.share. Vers une nouvelle génération de rétines programmables. In *GRETSI'99*, Nantes, France, September 1999.

[KIIY03] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida. A digital vision chip specialized for high-speed target tracking. *IEEE Trans. on Electron Devices*, 50(1) :191–199, 2003.

[KKI04] T. Komuro, S. Kagami, and M. Ishikawa. A dynamically reconfigurable simd processor for a vision chip. *IEEE journal of solid-state circuits*, 26(1) :265–269, January 2004.

[KMaJH+93] C. Koch, B. Mathur, S.-C. Liu ans J.G. Harris, J. Luo, and M. Sivilotti. *Adv. in Neural Information Processing Systems*, volume 5. In S.Hanson, J.Cowan, and L.Giles, editors, Morgan Kaufmann, San Mateo,CA, 1993.

[Kot96] U. Kothe. Local appropriate scale in morphological scale-space. *LNCS - Proc. of 4th European Conference on Computer Vision*, 1064 :219–228, 1996.

[KR89] T.Y. Kong and A. Rosenfeld. Digital topology : Introduction and survey. In *CVGIP*, volume 48, pages 357–394, 1989.

[Lag98] R. Laganière. A morphological operator for corner detection. *Pattern Recognition*, 31(11) :1643–1652, 1998.

[Lan78] C. Lantuejoul. *La squelettisation et son application aux mesures topologiques des mosaïques polycristallines.* PhD thesis, Ecole des Mines de Paris, 1978.

[LD95] S. Loncaric and A.P. Dhawan. Morphological signature transform and applications. 1995.

[LER95] L. Latecki, U. Elckhardt, and A. Rosenfeld. Well-composed sets. *Computer Vision and Image Understanding*, 61(1) :70–83, January 1995.

[LFE+99] G. Linan, P. Foldesy, S .Espejo, R .Dominguez-castro, and A .Rodriguez-Vasquez. A 0.5 $\mu$m cmos 106 transistors analog programmable array processor for real-time image processing. In *Proc. of the 1999 European Solid-State Circuits Conference*, pages 358–361, September 1999.

[LH02] B. Lee and M. Hedley. Background estimation for video surveillance. *IVCNZ02*, pages 315–320, 2002.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. 2(30) :79–116, 1998.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–649, 1981.

[LLMG99] L. Lacassagne, F. Lohier, M. Milgram, and P. Garda. Implémentation temps réel d'algorithme de détection de mouvement par champs de markov sur risc et dsp c6x. In *GRETSI'99*, 1999.

[LM00] S. Loncaric and T. Macan. Point-constrained optical flow for lv motion detection. In *Proceedings of SPIE Medical Imaging*, San Diego, USA, 2000.

[LMR97] J. Louchet, R. Mathurin, and B. Rottembourg. Combinatorial optimisation and linear prediction approaches to rain cell tracking. In *Proceedings od SPIE 1997*, Orlando, USA, 1997.

[Lou02] Jean Louchet. Introduction à la vision artificielle. ENSTA, 2002.

[Lyo81] R.L. Lyon. The optical mouse and an architectural methodology for smart digital sensors. In *Proc. of CMU Conference on VLSI Systems and Computations*, pages 1–19. Computer Science Press, 1981.

[MA99] T.G. Morris and Al. A column-based processing array for high-speed digital image processing. In *Conference on Advanced Research in VLSI*, pages 42–56, Atlanta, GA, 1999.

[Mah94] M.A. Mahowald. Analog vlsi chip for stereocorrespondence. In *Proc. IEEE Int. Symp. on Circuits and Systems*, london, UK, May-June 1994.

[Man00] A. Manzanera. *Vision artificielle rétinienne*. PhD thesis, Ecole Nationale Supérieure des Télecommunications, Novembre 2000.

[Man02] Antoine Manzanera. Cours de morphologie mathématique. ENSTA, 2002.

[Man03] Antoine Manzanera. Analyse d'images et vision - les espaces d'échelle en analyse d'images. ENSTA, 2003.

[MB90] F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1) :21–16, 1990.

[MB99] J. Monteil and A .Beghdadi. Une méthode repide et robuste d'estimation du champ de déplacement. In *Vision Interface'99*, Trois-Rivières, Canada, May 1999.

[MB02] A. Manzanera and T.M. Bernard. Mb : a coherent collection of 2d parallel thinning algorithms. Technical report, October 2002.

[MB03] A. Manzanera and T.M. Bernard. Metrical properties of a family of 2d parallel thinning algorithm. In *Proc. IWCIA - Electronic Notes on Discrete Mathematics*, volume 12, Palermo, Italy, May 2003. Elsevier.

[MBP98] D. Mercier, T.M. Bernard, and F. Paillet. L'âge de la maturité pour les rétines artificielles programmables. In *Actes XVIIème colloque ANRT Imagerie Rapide et Photonique*, Strasbourg, France, june 1998.

[MC80] C.A. Mead and L.A. Conway. *Introduction to VLSI Systems*. Addison-Wesley, Reading, Massachussets, 1980.

[Mea89] C.A. Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, Reading, Massachussets, 1989.

[MHC94] E. Memin, F. Heitz, and F. Charot. Efficient parallel non-linear multigrid relaxation algorithms for low-level vision applications. Technical report, Institut de recherche en informatique et systèmes aléatoires, 1994.

[Mj93] Mollon and jordan. *Study of women heterozygote for colour difficiency*. Vision research, 1993.

[MM88] C.A. Mead and A. Mahowald. A silicon model of early vision processing. *Neural Network*, (1) :91–97, 1988.

[MM99] F. Meyer and P. Maragos. Morphological scale-space representation with levelings. *LNCS - Scale-Space'99*, 1682 :187–198, 1999.

[Moi97a] A. Moini. An insect vision-based motion detection chip. *IEEE Journal of Solid-State Circuits*, 32(2) :279–284, February 1997.

[Moi97b] A. Moini. Vision chips or seeing silicon. Technical report, The University of Adelaide, 1997.

[Mor79] H.P. Moravec. Visual mapping by a robot rover. In *Proc. of the 7th IJCAI*, pages 598–600, 1979.

[MP04] A. Mittal and N. Paragios. Motion-based background subtraction using adaptative kernel density estimation. 2004.

[MS95a] R.E. Marston and J.C. Shih. Multi-scale skeletal representations of images via voronoi diagrams. *LNCS - Proc. of 4th European Conference on Computer Vision*, (1064) :219–228, 1995.

[MS95b] N. McFarlane and C. Schofield. Segmentation and tracking of piglets in images. *Machine Vision and Applications*, (8) :187–193, 1995.

[MS98] F. Mokhtarian and R. Suomela. Robust image corner detection through curvature scale space. *IEEE Trans. on Pattern Anlysis and Machine Intelligence*, 20(12) :1376–1381, 1998.

[MS01] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *ICCV*, Vancouver, Canada, July 2001.

[MY87] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic b-splines. *Computer Vision and Image Processing*, 39 :267–278, 1987.

[NCC02] D. Navarro, G. Cathebras, and G. Cambon. Rétine d'acquisition du flot optique : intégration de la transformée de census. In *Colloque du GDR CAO de Circuits et Systèmes Intégrés*, pages 93–96, 2002.

[Ngu96] P. Nguyen. *Machine de vision à rétine intelligente*. PhD thesis, Université Paris 11, 1996.

[NITM00] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno. 1ms column parallel vision system and it's application of high speed target tracking. In *Proc. IEEE Int. conf. on robotics and automation*, pages 650–655, San Francisco, CA, USA, April 2000.

[OB97] J.M. Odobez and P. Bouthemy. *Video data compression for multimedia computing*. Kluwer Academic Publisher, 1997.

[ONK94] J. Ohta, Y. Nitta, and K. Kyuma. Optical learning chip. 1994.

[OTJC02] F. Ortiz, F. Torres, E. De Juan, and N. Cuenca. Colour mathematical morphology for neural image analysis. *Real-Time Imaging*, 8 :455–465, 2002.

[Pai01] F. Paillet. *Intégration et évaluation de rétines artificielles numériques Programmables de hautes performances*. PhD thesis, Université Paris 6, Septembre 2001.

[Par93] N. Paris. Pompc :. a c language for data parallelism, 1993.

[Pav82] T. Pavlidis. *Algorithms for graphics and image processing*. Computer science press, 1982.

[PD94] A. Pikaz and I. Dinstein. Using simple decomposition for smoothing and feature point detection of noisy digital curves. 8(16) :808–813, 1994.

[PGO94] M. Proesmans, L. Van Gool, and A. Oosterlinck. Determination of optical flow and its discontinuities using a non-linear diffusion. pages 295–304, 1994.

[Pic04] M. Piccardi. Background subtraction techniques : a review. In *Proc. IEEE Conference on Computer*, February 2004.

[Pin03] A. Pinna. *Conception d'une rétine connexionniste : du capteur au système de vision sur puce*. PhD thesis, Université Paris 6, Décembre 2003.

[PMB98] F. Paillet, D. Mercier, and T.M. Bernard. Making the most of 15k lambda2 silicon area for a digital retina PE. In *Proc. SPIE, Vol. 3410, Advanced Focal Plane Arrays and Electronic Cameras*, Zrich, Switzerland, may 1998.

[PMB99] F. Paillet, D. Mercier, and T.M. Bernard. Second generation programmable artificial retina. In *Proc. IEEE ASIC/SOC Conf.*, pages 304–309, september 1999.

[Pra79] W. Pratt. *Digital image processing*. Wiley, NY, 1979.

[PVT93] T. Peli, L. Vincent, and V. Tom. Morphology-based algorithms for target detection/segmentation. In *SPIE on Architecture, Hardware, and Forward-Looking Infrared Isues in Automatic Target Recognition*, Orlando, USA, April 1993.

[RD03] C. Ranchin and F. Dibos. Moving objects segmentation using optical flow estimation. Technical report, UPD Ceremade, December 2003.

[RHKvH92] L. Rosenthaler, F. Heitger, O. Kübler, and R. v.d. Heydt. Detection of general edges and keypoints. In LNCS Springer, editor, *proc. 2nd European Conf. on Computer Vision*, number 588, pages 78–86, 1992.

[RK06] V. Rodenhorst and A. Koschan. Comparison and evaluation of feature point detectors. Berlin, Germany, March 2006.

[Ros58] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. 65(6) :386–408, 1958.

[Ros62] F. Rosenblatt. *Principles of Neurodynamics*. Spartan Books, 1962.

[RUV68] R.G. Runge, M. Uemura, and S.S. Viglione. Electronic synthesis of the neural networks in the pigeon retina. In *Cybernetic problems in bionics*, pages 791–800, Dayton, Ohio, USA, May 1968.

[Sal02] P. Salembier. On filters by reconstruction for size and motion simplification. In *Sixth International Symposium on Mathematical Morphology*, pages 425–434, April 2002.

[SB97] S.M. Smith and J.M. Brady. Susan - a new approach to low level image processing. *Int. Journal on Computer Vision*, 23(1) :45–78, 1997.

[SBB+92] E. Sackinger, B.E.Boser, J. Bromley, Y. LeCun, and L.D. Jackel. Application of the anna neural network chip to high-speed character. *IEEE Trans. on computer*, 3(3) :498–505, 1992.

[Ser82] Jean Serra. *Image analysis and mathematical morphology*. Academic Press, London, 1982.

[Ser88] Jean Serra. *Image analysis and mathematical morphology*, volume 2. Academic Press, London, 1988.

[Ser00] Jean Serra. Cours de morphologie mathématique. Ecole des Mines de Paris, 2000.

[SG99] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR99*, Fort Colins, CO, USA, June 1999.

[SG00] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real time tracking. *IEEE trans. on PAMI*, 8(22) :747–757, August 2000.

[SL03] N. Sebe and M.S. Lew. Comparing salient point detectors. *Pattern recognition letters*, 24 :89–96, 2003.

[SM99] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 10(11), 1999.

[SMB] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. Technical report, INRIA.

[SMB98] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. In *ICCV*, Bombay, India, January 1998.

[SMK02] S. Sarkar, D. Majchrzak, and K. Korimilli. Perceptual organization based computional model for robust segmentation of moving objects. *Computer Vision and Image Understanding*, (86) :141–170, February 2002.

[SOG98] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE trans. on Image Processing*, 7(4) :555–580, April 1998.

[Sor94] Y. Sorel. "massively parallel systems with real time constraints, the "algorithm architecture adequation methodology"". In *int. Conf. on Massively Parallel Computing Systems*, Ischia, Italy, may 1994.

[Sud01] A. Sud. Estimation of motion field. In *Computer Vision*, 2001.

[SWG97] E. Shilat, M. Werman, and Y. Gdalyahu. Ridge's corner detection and correspondance. In *Proc. of the Conf. on Computer Vision and Pattern Recognition*, pages 976–981, Puerto Rico, USA, 1997.

[TC89] C-H. Teh and R.T. Chin. On the detection of dominants points on digital curves. *IEEE trans. on Pattern Analysis and Machine Intelligence*, 11(8) :859–872, August 1989.

[TH99a] A. Torralba and J. Herault. An efficient neuromorphic analog network for motion estimation. *IEEE Trans. on circuits and systems-I : special issue on bio-inspired processors and CNNs for vision*, 46(2), February 1999.

[TH99b] A.B. Torralba and J. Herault. Asymmetrical filters for vision chips : a basis for the design of large sets of spatial and spatiotemporal filters. In *7th Int. Conf. on Micro-electonics for Neural, Fuzzy and Bio-inspired Systems*, pages 224–231, Granada, Spain, 1999.

[TK01] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, 2001.

[TKBM99] K. Toyoma, J. Krumm, B. Brumitt, and B. Meyers. Wallflower : principles and practice of background maintenance. In *Proc. IEEE Conference on Computer*, pages 255–261, Kerkyra, Greece, 1999.

[Vin93] L. Vincent. Morphological grayscale reconstruction in image analysis : applications and efficient algorithms. *IEEE trans. on Image Processing*, 2 :176–201, April 1993.

[WADP97] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder : Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7) :780–785, 1997.

[Wil91] S.S. Wilson. Teaching network connectivity using dimulated annealing on a massively parallel processor. volume 19(4), pages 559–566, 1991.

[WL97] D. Wang and C. Labit. Morphological spatio-temporal simplification for video image segmentation. *Signal Processing. Image Communication*, 11 :161–170, 1997.

[WT92] R-Y. Wu and W-H. Tsai. A new one-pass parallel thinning algorithm for binary images. *Pattern Recognition Letters*, 13 :715–723, 1992.

[Xa96] X.Arreguit and al. A cmos motion detector system for pointing devices. *IEEE Journal of Solid-State Circuits*, 31(12) :1916–1921, December 1996.

[YTF75] S. Yokoi, J. Torikawi, and T. Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Computer Graphics and Image Processing*, 4 :63–73, 1975.

[Zav81] B. Zavidovique. *Contribution à la vision des robots*. PhD thesis, Université de Besançon, 1981.

[ZBM88] B. Zavidovique, P. Bonnin, and C. Millour. Méthodes de détection de mouvement - applications à la poursuite. In *(Confrence Invite) Ecole d'automne Thomson*, JOUY-EN-JOSAS, 1988.

[ZZ95] X. Zhang and D. Zhao. A morphological algorithm for detecting dominant points on digital curves. *SPIE Proc. Non Linear Image Processing VI*, 2424 :372–383, 1995.

[rk91] M. rkisz. Localisation d'objets mobiles dans des scènes naturelles filmées par une caméra fixe. *Traitement du signal*, 9(4) :325–346, September 1991.

# Publications de l'auteur

- A. Manzanera et J. Richefeu, "*A new motion detection algorithm based on* Σ-Δ *background estimation*", soumis à Pattern Recognition Letters.
- J. Richefeu, "*Rapport d'activité de monitorat 2002-2005*", Juin, 2005.
- J. Richefeu et A. Manzanera, "*Détection de mouvement par capteur intelligent*", ORASIS, Clermont-Ferrand, Mai, 2005.
- A. Manzanera et J. Richefeu, "*A robust and computationally efficient motion detection algorithm based on* Σ-Δ *background estimation*", ICVGIP, Kolkata, India Décembre, 2004.
- J. Richefeu et A. Manzanera, "*A new hybrid differential filter for motion detection*", ICCVG, Warsaw, Poland, Octobre, 2004.
- J. Richefeu et A. Manzanera, "*Morphological dominant points detection for motion analysis on programmable retina*", ISSPA, Paris, France, Juillet, 2003.
- J. Richefeu, "*Dominant points detection and its cellular implementation*", ISSPA, Paris, France, Juillet, 2003.
- J. Richefeu, "*Détection / Reconnaissance d'objets sur rétines artificielles mixtes synchrones / asynchrones*", rapport de DEA, Septembre 2002.

# Index