REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE

Departement d'Electronique

Projet de Fin D'Etudes

En vue de l'obtention du Diplome
d'Ingénieur d'Etat en Electronique

Theme

Detection du Mouvement dans une Séquence Video par Filtres Morphologiques

Proposé et dirigé par :

- M. C.LARBES
- MME. S.AIT DAOUD

Réalisé par :

- M. LAOUAR NIZAR
- M. LARABA MOHAMMED SEDDIK

Promotion : Juin 2009

المُلَخَّص

هذَا العَمَل يَتَمَثَّل فِي وَسـم خَوَارِزمِيَّات كَشف الحَرَكَة فِي مَقطَع مِن فِلم ، مَعَ العِلم أَنَّ الطُرُق المُختَارَة: المُتَوَسِط التَرَاجُعِي، التَدَرُج الزَمَنِي، التَدَرُج المَنسِي، المُـقَـيِّم $\Sigma - \Delta$. وَلَقَد أَنـجَـزنَا وَاجِهَة تَصوِيرِيَة فِي MATLAB تَسمَح بِمُعَايَنَة مُختَلَف النَتَائج المُتَحَصَّل عَلَيهَا. وَبَعدَهَا قُمـنَا بِوَسـم التَدَرُج الزَمَنِي، المَنـسِي وَ المُتَوَسِط التَرَاجُعِي عَلَى اللَّوحَة RC203E، وَ الَّتِي تَـحتَـوِي عَلَى الدَارَة FPGA Xilinx Virtex II.

لُغَة الوَصف المُختَارَة هِيَ Handle-C نَظَرًا لِسُـهُولَتِهَا.

كَلِمَات مِفتَاحِيَة: المُعَالجَات التَشكِيلِيَة (المُورفُولُوجِيَة)، المُتَوَسِط التَرَاجُعِي،$\Sigma - \Delta$، Handle-C، FPGA Xilinx Virtex II ، اللَّوحَة RC203E.

Résumé

Ce travail consiste à implémenter quelques algorithmes de détection du mouvement dans une séquence vidéo à savoir ; la méthode basée sur la moyenne récursive, le gradient temporel, le gradient oublieux et l'approche $\Sigma - \Delta$. Nous avons donc réalisé une interface graphique en MATLAB qui permet de visualiser les différents résultats obtenus. Par la suite, nous avons implanté le gradient temporel, oublieux et la moyenne récursive sur la carte RC203E contenant le circuit FPGA Xilinx Virtex II. Le langage de description choisi est le Handle-C pour sa simplicité.

Mots clés : Filtres morphologiques, moyenne récursive, $\Sigma - \Delta$, FPGA Xilinx Virtex II, Handle-C, la carte RC203E

Abstract

This work consists of an implementation of some algorithms of motion detection in a video sequence with knowing ; the method based on the recursive average, the temporal gradient, the forgetting gradient and the approach $\Sigma - \Delta$, we thus made a graphic interface in MATLAB which makes it possible to display the various results obtained. Thereafter, we established the temporal, forgetting gradient and the recursive average on the board RC203E containing the circuit FPGA Xilinx Virtex II. The selected language of description is Handle-C for its simplicity.

Key words : Morphological filters, recursive average, $\Sigma - \Delta$, FPGA Xilinx Virtex II, Handle-C, the board RC203E

REMERCIEMENT

Nous tenons à exprimer notre reconnaissance à Mr *C.LARBES* et Mme *S.AITDAOUD*, qui nous ont donné l'occasion de travailler sur un sujet si passionnant, et pour leur confiance, leurs conseils et leur aide précieuse.

Nous remercions profondément Mr *M.S.AITCHEIKH* de nous avoir fait l'honneur de présider le jury.

Nos remerciements vont aussi à Mr *B.BOUSSEKSOU* pour avoir accepter d'examiner ce modeste travail.

Notre profonde gratitude à toutes les personnes ayant contribué à notre formation.

A mes très chers parents
A mes frères et ma soeur
A mon oncle Bachir
A mon cousin Walid
A toute la famille
et à tous mes amis

éddik

A mes chers parents
A mes chers frères et soeurs
Per tutta la famiglia
A tous mes amis et mes camarades de classe
Aux enfants de Gaza

L.N

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

La détection et le calcul du mouvement dans une séquence vidéo sont très utilisés de nos jours et particulièrement en télésurveillance, en médecine, poursuite des cibles, compression des images animées, analyse et reconnaissance des objets dans une séquence d'images ...etc. Les méthodes fréquemment appliquées dépendent étroitement de la variation de la luminance (intensité du pixel), on peut citer : les méthodes fréquentielles, les méthodes différentielles, la corrélation de blocs ... Or, en réalité la luminance n'est jamais constante dans le temps (variation de l'éclairage, brouillard, ...). Pour résoudre ce problème, nous allons présenter quelques approches différentes basées sur les filtres morphologiques et le calcul de l'image du fond statique. Cette dernière est adaptative et est mise à jour à chaque fois qu'on introduit un nouvel objet statique. Ce rapport est composé de deux parties :

Une partie théorique, qui comporte les chapitres suivants :

- Le chapitre I comporte quelques notions de base sur le traitement d'images en général, et les algorithmes de prétraitement nécessaires pour améliorer la qualité de l'image.
- Le chapitre II est consacré aux algorithmes de détection de mouvement. En adoptant des approches fréquentielles, différentielles ainsi qu'un algorithme de corrélation de blocs.
- Dans le chapitre III, nous explicitons de plus proche les caractéristiques des méthodes morphologiques.
- Dans chapitre IV, nous donnons des généralités sur les circuits FPGA .

La seconde partie (conception et réalisation), contient les chapitres suivants :

- Le chapitre V, qui est relatif à l'étude de la carte RC203E , où on donne une description de l'environnement de développement, le langage utilisé ainsi que l'implémentation des algorithmes des méthodes morphologiques citées dans le chapitre III.
- Dans le chapitre VI nous allons présenter et comparer les résultats obtenus pour chaque méthode, qui nous les illustrerons dans une interface graphique, cette dernière qui contient tout les scripts qu'on a mis au point durant notre travail.

On a attaché à ce rapport une annexe qui comporte les scripts MATLAB de certains algorithmes qu'on a jugé utile de les mettre afin qu'ils soient une matière première de développement.

Première partie

Etude théorique

1

Généralités sur le traitement d'images

1.1 Introduction

Dans ce chapitre nous allons donner des généralités détaillées des concepts de base du traitement d'images, en commençant par les types d'images ainsi qu'une définition explicite sur la représentation des images numériques en énumérant les différentes caractéristiques du traitement (acquisition d'une image, les dimensions et la résolution). On donne aussi les différents paramètres qui caractérisent les images particulièrement la luminance et le contraste, puis nous allons donner les différents types de bruit qu'on trouve dans une image ainsi que leurs source de provenance. Enfin nous allons parler brièvement sur les filtres (linéaires et non-linéaires) utilisés ainsi leurs impacts sur les différents types de bruit.

1.2 L'image numérique

Contrairement aux images obtenues à l'aide d'un appareil photo, ou dessinés sur du papier, les images manipulées par un calculateur (ordinateur) sont *numériques* (représentées par une

série de bits).

L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevées à l'emplacement correspondant dans l'image réelle, ou calculées à partir d'une description interne de la scène à représenter.

Il existe deux sortes d'images numériques : les images matricielles et les images vectorielles.

Les images vectorielles Dans une image vectorielle les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique, par exemple, un cercle est décrit par une information du type | cercle | position du centre | rayon | . Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans. Elles présentent deux avantages :

- Occupent peu de place en mémoire
- Peuvent être redimensionnées sans perte d'information.

Les images matricielles Une image matricielle est formée d'un tableau de n colonnes et m lignes de points ou pixels. Plus la densité des points est élevée, plus le nombre d'informations est grand et plus la résolution de l'image est élevée. Corrélativement la place occupée en mémoire et la durée de traitement seront d'autant plus grandes. Les images vues sur un écran de télévision ou une photographie sont des images matricielles. On obtient également des images matricielles à l'aide d'un appareil photo numérique, d'une caméra vidéo numérique ou d'un scanner.

$$I = \begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,n-1} \\ P_{1,0} & \ddots & & P_{1,n-1} \\ \vdots & & \ddots & \vdots \\ P_{m-1,0} & P_{m-1,1} & \cdots & P_{m-1,n-1} \end{pmatrix}$$

$P_{i,j}$ est le pixel de la ième ligne et de la jème colonne

FIG. 1.1 – Image en pixels

1.3 La représentation d'une image

L'image fait partie des signaux informationnels telle que la parole, considérée comme une projection d'objet tridimensionnelle, elle est représentée par une fonction à deux dimensions $f(x, y)$. L'amplitude de f à toute paire de coordonnées (x, y) est appelée l'intensité de l'image à ce point[RT06].

Les images qu'on va utiliser par la suite, sont des images matricielles de nature bidimensionnelle discrète. A chaque élément (*pixel*) de l'image, correspond une intensité de luminance ou de chrominance (*couleur*). La plus grande intensité correspond à la couleur blanche, et la plus petite correspond à la couleur noire.

1.3.1 Le pixel

L'image est constituée d'un ensemble de petits éléments appelés *pixels*, (*pixel* est l'abréviation de **pic**ture **el**ement, c'est-à-dire qu'il s'agit du plus petit élément constitutif d'une image). C'est une entité calculable qui peut recevoir une structure et une quantification. La quantité d'information qui véhicule chaque pixel donne des nuances entre les images monochromes et les images en couleurs. Dans le cas d'une image monochrome, chaque pixel est codé sur un seul canal. Dans une image couleur **RVB**[1] , un pixel peut être représenté sur trois canaux, un pour chacune des couleurs.

En général on associe à chaque canal un seul *octet*, donc pour une image monochrome (à niveaux de gris) chaque pixel est codé sur un seul octet, d'où il a 2^8 (256) valeurs possibles,

[1] R(Rouge), V(Vert) et B(Bleu)

entre 0 (noir) et 255 (blanc). Pour les images couleurs le pixel est codé sur trois octets, ce qui donne 2^{24} valeurs possibles.

1.3.2 La résolution

La résolution est la clarté ou la finesse de détails d'une image numérique, d'où on peut la définir comme étant le nombre de pixels par unité de longueur. On utilise aussi le mot résolution pour désigner le nombre total de pixels affichables horizontalement et verticalement sur un moniteur, par exemple pour 1000 pixel horizontalement et 1000 pixel verticalement on a une résolution de 1 M.pixel[(2)] ; plus grand est ce nombre, meilleure est la résolution. En général on utilise l'unité de longueur anglo-saxonne le *pouce* ou l'*inch*. La résolution d'une image s'exprime alors en pixels par pouce (ppp) ou dots per inch (dpi)[RT06].

1.3.3 Les dimensions

Les dimensions sont la hauteur et la largeur de l'image exprimées en pixel, donc le produit de ces deux grandeurs nous donnera le nombre de pixels de cette image.

1.3.4 Le voisinage d'un pixel [PC95]

L'image est représentée généralement par un maillage carré, ainsi les métriques utilisées le plus souvent en maillage carré sont d_4 et d_8 définies par :

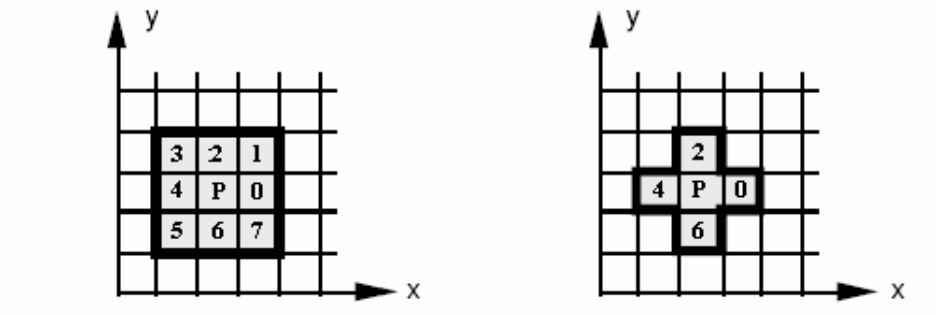
1. $d_4(p,q) = |i_p - i_q| + |j_p - j_q|$ tel que i_p, j_p sont respectivement l'indice de la ligne et de la colonne du pixel p, donc on appelle un voisinage 4-connexe le voisinage v_4 tel que :
   $v_4 = \{q \in I; d_4(p,q) \leq 1\}$ avec
   I : l'image ; p : le pixel courant ; q : l'ensemble de pixels qui appartiennent au voisinage v_4
2. $d_8(p,q) = MAX(|i_p - i_q|, |j_p - j_q|)$ tel que i_p, j_p sont respectivement l'indice de la ligne et de la colonne du pixel p, donc on appelle un voisinage 8-connexe le voisinage v_8 tel

[2]M.pixel : Million pixel

que :

$v_8 = \{q \in I; d_8(p,q) \leq 1\}$ avec

I : l'image ; p : le pixel courant ; q : l'ensemble de pixels qui appartiennent au voisinage v_8



a) 8-connexe b) 4-connexe

FIG. 1.2 – Voisinage d'un pixel

1.3.5 La connexité

La connexité entre les pixels d'une image est un concept très important utilisé surtout dans l'établissement des frontières des objets et l'identification des composants d'une région dans une image. Pour établir si deux pixels sont connectés, nous devons déterminer s'ils sont adjacents quelques parts (par exemple, s'ils appartiennent à un voisinage 4-connexe) et si leurs niveaux de gris respectent un certain critère de similarité.

1.4 Caractéristiques d'une image

1.4.1 Luminance

L'intensité ou luminance est le caractère qui indique l'intensité de lumière perçue indépendamment de la couleur. Elle s'étend du noir au blanc avec toutes les nuances de gris si on ne voit pas la couleur.

1.4.2 Contraste

C'est la différence marquée entre deux zones d'une image, plus précisément c'est le taux entre les zones sombres et les zones claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images. Si $L1$ et $L2$ sont les degrés de luminosité respectivement de deux zones voisines $A1$ et $A2$ d'une image, le contraste C est défini par le rapport [RT06] :

$$C = \frac{L1 - L2}{L1 + L2}$$

1.4.3 Saturation

La saturation (la pureté) correspond à la pureté de la couleur d'une surface jugée en proportion de sa brillance. La saturation va du gris neutre aux couleurs pastels, jusqu'aux couleurs saturées. Plus une densité spectrale est concentrée autour d'une longueur d'onde dominante, plus la couleur associée est saturée. Il est inversement possible de nuire à une couleur en ajoutant une lumière qui possède une puissance non nulle sur tout le spectre.

C'est ce qui nous fait distinguer une couleur vive de la délavée , une couleur vive est une couleur saturée à 100 %.

1.5 L'acquisition d'une image

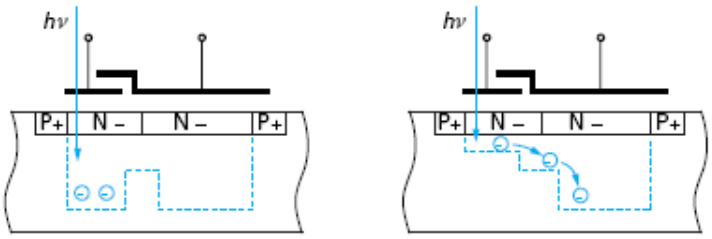
Différents types de capteurs sont disponibles pour générer des images. Ils se distinguent par : leur principe d'acquisition, leur vitesse d'acquisition, leur résolution spatiale, leur gamme spectrale ou encore leur dynamique. Néanmoins, les capteurs utilisés le plus fréquemment dans les systèmes de vision industriels sont les circuits à transfert de charges CCD. Ils découlent de l'association d'une cellule photosensible et d'un dispositif de transfert de charge. Notons que ces capteurs réalisent l'échantillonnage de l'image, mais pas sa quantification.

Les dispositifs CCD se présentent sous la forme d'une série de capacités MOS (metaloxide semiconductor) couplées, constituant ainsi des registres analogiques où les charges électriques

représentant l'information à transmettre sont stockées puis décalées en série vers la sortie du dispositif. Ainsi, l'intégration dans une même puce électronique de séries de photo-éléments associés à des registres analogiques CCD permet d'obtenir des capteurs d'images à l'état solide. A l'aide de ces circuits, l'image est acquise en deux temps :

- un temps d'intégration.
- un temps de transfert.

Durant le temps d'intégration, la lumière incidente, d'énergie $h\nu$, est absorbée par les photo-éléments puis transformée en paquets de charges électriques proportionnelles à l'intensité lumineuse incidente. Les paquets de charges sont alors transférés vers les registres analogiques via une porte de transfert et décalés en série vers la sortie du CCD [PAI01].



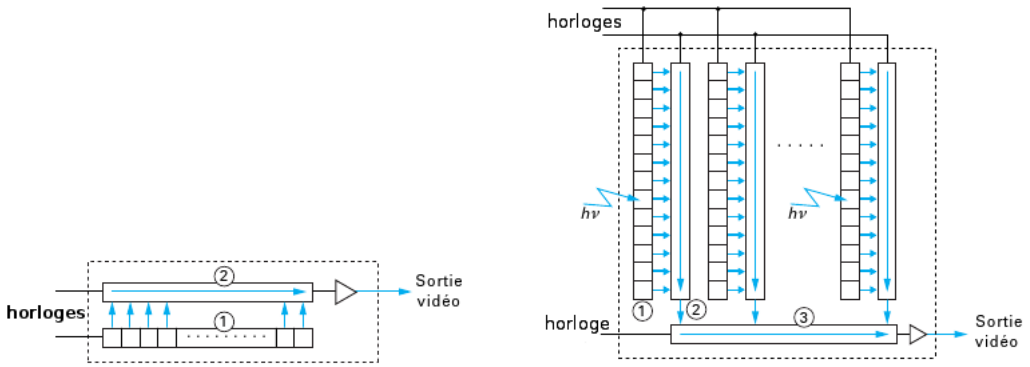
a) Acquisition b) Transfert

FIG. 1.3 – Acquisition et transfert d'une image par un capteur CCD

Ces photo-éléments sont organisées par deux façons, d'où les deux types de capteurs CCD.

Capteurs linéaires Comme indique leurs nom, les éléments photosensibles sont donc rangés dans une seule ligne, d'où pour pouvoir acquérir une image, il faut que cette ligne se déplace perpendiculairement par rapport à elle même et avec une vitesse constante.

Capteurs Matriciels Les photo-éléments dans ces capteurs sont organisés matriciellement, donc dans ce cas on n'a pas besoin d'un déplacement, de plus les registres doivent combiner la transmission des lignes avec celle des colonnes.



a) capteur linéaire b) capteur matriciel

FIG. 1.4 – Les deux types des capteurs CCD

Pour la figure FIG 1.4 : (1) photo-éléments, (2) et (3) Registres à décalage

1.6 Processus d'analyse d'image

Le processus d'analyse d'image a pour but de fournir une description ou une interprétation d'une scène à partir de l'information extraite de l'image. Il peut être structuré en plusieurs étapes, comme le montre la figure ci-dessous :

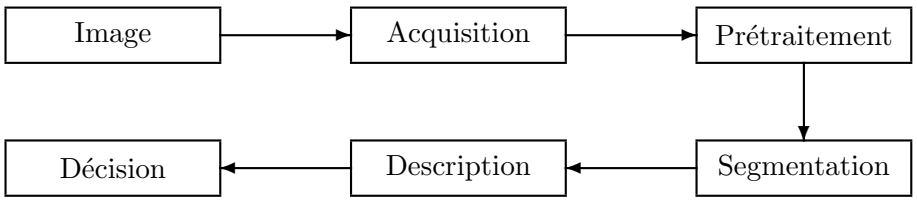


FIG. 1.5 – Les étapes du processus d'analyse d'images

Tout d'abord on doit commencer par l'*acquisition*, c'est la discrétisation de notre image continue, mais en générale la quantité d'informations qui porte l'image discrète est très volumineuse et difficile à manipuler, de même l'acquisition entraîne des pertes d'informations. Pour les images bruitées ou floues, on doit passer par un *prétraitement* avant toute opération

de détection, où les informations dégradées sont restaurés. Ensuite vient l'étape la plus difficile (la *Segmentation*), qui correspond à trouver les régions de l'image qui ont un sens. A partir de là, viennent le traitement du *haut niveau*, telle que la *description* de l'image qui peut être par exemple une reconnaissance de formes, et les *décisions* là où on commande notre système, ces deux dernières étapes peuvent être prises à partir des résultats fournis par la segmentation.

1.7 Le seuillage

Une étape clé du traitement de l'image est le seuillage c'est à dire le passage de l'image numérique à l'image binaire. Les traitements binaires sont destinés à définir le plus précisément possible les objets à analyser.

FIG. 1.6 – Seuillage d'une image $\delta = 118$

$$S(x,y) = \begin{cases} 0 & si \quad I(x,y) \leq \delta \\ 255 & sinon \end{cases}$$

δ : le seuil, $I(x,y)$: l'image originale, $S(x,y)$: l'image seuillée.

1.8 Le Bruit

En traitement du signal, le bruit se définit comme étant toute sorte de signal qui parasite notre signal utile, en traitement d'image toute brusque fluctuation d'un pixel par rapport à ses

voisins étant considéré comme un bruit que se soit fluctuation de luminance ou de contraste.

Les facteurs qui font provenir ce bruit sont :

Bruit lié aux dispositifs d'acquisition La caméra, les câbles et tout autre bloc de connexion engendre un bruit qui à un effet multiplicatif, pour cela il serait plus judicieux de mettre les sources qui produisent plus de bruit dans les étages inférieurs pour diminuer leurs contribution au bruit.

Les capteurs optiques aussi peuvent introduire des bruits de ce genre, s'il sont défectueux ou s'il ne travaillent pas dans la zone linéaire (appelé souvent *zone de mesure nominale*), c'est à dire soit la zone de non détérioration soit la zone de déformation,

De ce fait, nous pouvons ainsi obtenir une distorsion de la gamme des niveaux de gris, provoquant une saturation ou bien une distorsion géométrique de l'image équivalente à l'effet d'un miroir grossissant par exemple, des flous . . . *etc.*

Pour restaurer l'image convenablement il faut agir au niveau du capteur directement. Nous pouvons aussi construire un modèle a priori du phénomène de dégradation ; une inversion systématique est alors envisageable comme dans le cas de la microscopie électronique où il est très fastidieux d'obtenir des images de bonne qualité, mais les phénomènes dégradants produisant le plus souvent un mauvais contraste ou une dérive lumineuse sont envisageable et maîtrisés.

Bruit lié à la scène Il se trouve parfois que la scène soit polluée de poussière, ou couverte de nuage, brouillard ou buée qui dégrade la qualité des images . De même, la présence des os au niveau de la cage thoracique perturbe le processus de radiographie. Cependant, là encore, la connaissance a priori du phénomène perturbateur permet d'envisager une modélisation et donc une correction systématique. Ce type de bruit a un effet de bruit additif.

Bruit lié à la transmodulation Les images qui sont capturés à partir d'une séquence vidéo qui regroupe son et image dans un même signal, sont atteintes par un bruit dû à la transmodulation qui se produit du fait qu'on utilise une double modulations (modulation audio et modulation vidéo), dans le cas où des composantes du signal audio bruitent notre signal vidéo

Bruit lié à l'échantillonnage Une image est un signal discret. Il est donc nécessaire de passer du domaine continu au domaine discret, au niveau de la scène et au niveau de l'intensité lumineuse émanant de celle ci. De manière générale, le spectre des intensités lumineuses noir et blanc est quantifié sur 256 niveaux de gris différents si on admet que chaque pixel est codé sur un octet.

Il faut savoir que la précision des dispositifs électronique est supérieure à ce que l'œil humain utilise dans ses tâches quotidiennes. Par contre, la quantification de l'espace en $(n \times p)$ pixels est un inconvénient majeur ; les capteurs actuels ne permettent pas d'obtenir une précision satisfaisante pour permettre un contrôle dimensionnel de formes.

Néanmoins, tout objet dont la taille est inférieure à l'unité de tessellation du plan disparaît ou bien n'est représenté que par un pixel. Cet effet est connu sous le nom d'effet *"poivre et sel"*. Ce bruit est aussi généré par une texture dont les caractéristiques sont trop fines par rapport à la fréquence d'échantillonnage (phénomène de Moiré).

Ce sont les sources les plus fréquentes de dégradation de la qualité d'image. Pour rendre cette dernière plus nette, on utilise les techniques de filtrage, qui seront détaillées dans les paragraphes suivants.

1.9 Le filtrage

On fait appel aux différentes techniques de filtrage afin de donner une harmonie à l'ensemble de l'image par l'élimination du bruit qu'on vient de le définir comme étant une brusque fluctuations d'un pixel par rapport à son voisinage et donc on utilise ces derniers pour imposer le pixel en question. parmi ces techniques de filtrage on cite :

- Le filtrage *linéaire*
- Le filtrage *non linéaire*
- Le filtrage *homomorphique*
- Le filtrage *morphologique*

1.9.1 Le filtrage linéaire[RT06]

Ces filtres manipulent le pixel suivant ces pixels voisins en lui attribuant une nouvelle valeur qui n'est autre qu'une combinaison linéaire du voisinage y compris le pixel en question.

Filtre moyenneur

Ce processus s'élabore grâce à un masque H qui est en réalité une matrice unitaire[3] $n \times n$ multiplié par $\frac{1}{n^2}$.

Dans la majorité des cas il s'agit d'une matrice 3×3. elle opère donc sur le pixel central via les huit pixel qui l'entoure et ceci s'effectue bien par le produit de convolution.

$$H = \frac{1}{n^2} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

L'image finale filtrée est une convolution de l'image avant le traitement avec le masque qui balaye toute l'image en se déplaçant sur celle ci, malheureusement les pixels qui composent la

[3] tous les éléments sont égale à 1

périphérie de l'image ne seront pas modifié puisque ils ne possèdent pas huit pixels adjacents.

$$I_f(x,y) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} H(i,j) \times I_i(x+i-E[\frac{n}{2}], y+j-E[\frac{n}{2}])$$

où $E(x)$ est la partie entière de la variable x ; n est la taille du masque.

pour $n = 3$ on a :

$$I_f(x,y) = \sum_{i=0}^{2}\sum_{j=0}^{2} H(i,j) \times I_i(x+i-1, y+j-1)$$

Exemple prenons l'image à niveaux de gris 4×4 suivante

35	31	29	37
46	38	21	30
65	42	60	30
31	32	29	25

Après filtrage ⇒

35	31	29	37
46	**41**	**35**	30
65	**40**	**34**	30
31	32	29	25

a)Image originale *b)filtre* 3×3 *c)filtre* 9×9

FIG. 1.7 – Le filtre moyenneur

Filtre Smooth

On obtient ce filtre en utilisant la distribution gaussienne, d'où sa réponse impulsionnelle sera de la forme :

$$h(l,k) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{k^2+l^2}{2\sigma^2}\right)$$

Ce filtre est séparable car on peut écrire $h(k,l) = h_x(k) \cdot h_y(l)$, ceci nous permettra de calculer juste une colonne et puis la multiplier par sa transposée, comme par *exemple* pour un masque

3×3 et $\sigma = 1$:

$k, l = -1, 0, 1$;

$$h_x(k) = \left(\exp\frac{-1}{2}, \exp\frac{0}{2}, \exp\frac{-1}{2}\right)^T = (0.6065, 1, 0.6065)^T = h_y(l)^T$$

$$h(k,l) = h_x(k) \cdot h_y(l) \simeq \frac{1}{15}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

remarque : La grandeur $\frac{1}{2\pi\sigma^2}$ n'est pas utilisée car le somme des élément du filtre doit être égale à 1.

A^T : est la matrice transposée de A.

1.9.2 Le filtrage non linéaire

Ces filtres ont été développés pour combler les inconvénients que présentent les filtres linéaires, le filtre non linéaire s'oppose au précédent dans sa dénomination car il n'est pas le résultat d'une combinaison linéaire de pixels, les pixels voisins interviennent suivant une loi non linéaire.

Filtre médian

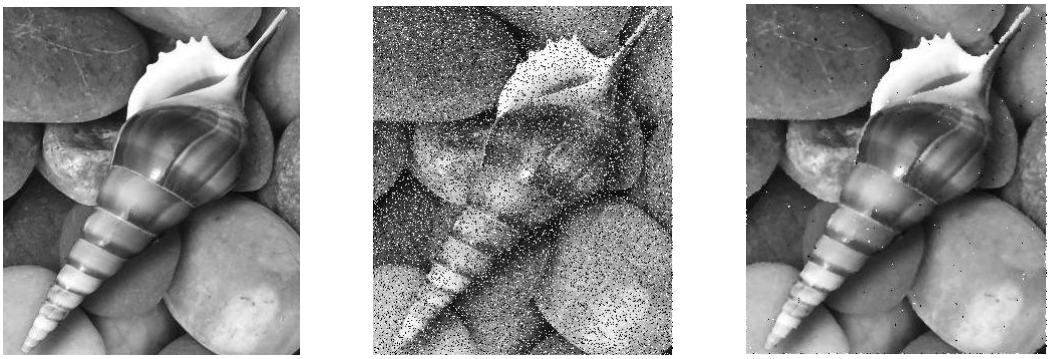
La médiane est une mesure statistique qui représente une alternative robuste à la moyenne. Considérons n valeurs numériques $x_1, ..., x_n$ (pas nécessairement distinctes), ou n est impair. On les ordonne d'une façon croissante afin d'avoir la valeur médiane qui est au milieu de cette suite : $y_m = \frac{(n+1)}{2}$

Alors que le filtre moyenneur introduit des flous sur le bord des objets, le filtre médian permet l'élimination des parasites isolés dans l'image sans affecter les contours. Il faut souligner l'intérêt du filtre médian qui est :

- Un pixel non représentatif dans le voisinage affectera peu la valeur médiane.
- La valeur médiane choisie étant le niveau de gris d'un des pixels considérés, nous ne créons pas alors de nouveaux niveaux de gris dans l'image. Ainsi, lorsque le filtre passe

sur un contour très marqué il le préservera mieux. Le filtre médian garde donc la netteté de l'image pour les éléments de dimensions importantes par rapport au noyau du filtre, mais élimine les détails fins de manière irrémédiable.

par exemple :



a)Image originale *b)image bruitée (sel et poivre)* *c)filtre* 3×3

FIG. 1.8 – Le filtre médian

1.9.3 Le filtrage Homomorphique

Le filtrage homomorphique est réalisé par une combinaison de traitements linéaire. Il permet souvent de se ramener à un problème de lissage de bruit additif [PC95]. Le schéma de principe d'un filtre homomorphique est :

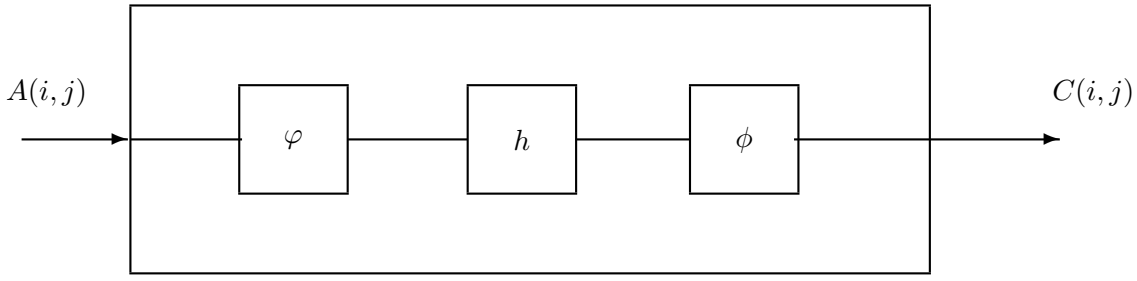


FIG. 1.9 – Schémas de principe du filtrage homomorphique

φ et ϕ sont deux filtres non-linéaires réciproques, h est la réponse impulsionnelle d'un filtre linéaire pass-bas en général.

1.9.4 Le filtrage morphologique

Le filtrage morphologique repose sur la morphologie mathématique, fondée sur une description ensembliste des images. Contrairement aux opérateurs précédemment présentés, on privilégie la notion de forme par rapport aux informations sur les amplitudes des variations d'intensité. A la place des opérations d'addition et de multiplication, les fonctions de base sont le minimum et le maximum. Ce type de filtrage est utilisé pour éliminer des pixels isolés dans les images binaires à deux niveaux de gris, qui sont considérés comme un bruit. Il met en correspondance chaque pixel et son voisin par une fonction logique (ET, OU). Parmi les opérateurs morphologiques, nous citons :

- *L'érosion*
- *La dilatation*
- *L'ouverture*
- *La fermeture*

Le filtrage par érosion

L'érosion permet d'éliminer les pixels blancs isolés. On effectue le ET logique des huit voisins du pixel considéré. Elle consiste en le choix du plus petit élément du masque, elle élimine les taches blanches dans les zones noires, mais ajoute des pixels noirs au contour des objets présents dans l'image.

Le filtrage par dilatation

La dilatation permet d'éliminer les pixels noirs isolés au milieu des parties blanches de l'image, on effectue le OU logique des huit voisins du pixel considéré.

En appliquant une dilatation, ces taches noires peuvent être éliminées, mais la taille des objets présents dans l'image diminue car la dilatation enlève des pixels du contour, entraînant une déformation de certains objets. La dilatation dépend du choix du plus grand élément de

a)Image originale *b)filtre* 3×3 *c)filtre* 13×13

FIG. 1.10 – Le filtrage par érosion

son masque, elle entraîne des effets habituels tels que : la séparation des objets à l'endroit des étranglements, le rétrécissement des objets de grande taille et la disparition des petites composantes . on prend l'exemple d'un pixel. donc la nouvelle valeur du pixel central est de 128 car elle est la plus petite valeur.

a)Image originale *b)filtre* 3×3 *c)filtre* 13×13

FIG. 1.11 – Le filtrage par dilatation

Le filtrage par ouverture

L'ouverture est constituée par une opération d'érosion suivie d'une dilatation. Elle permet de retrouver la taille normale des objets de l'image.

a)Image originale *b)filtre* 3×3 *c)filtre* 9×9

FIG. 1.12 – Le filtrage par ouverture

Le filtrage par fermeture

La fermeture est une opération morphologique qui consiste à faire subir à l'image une opération de dilatation suivie d'une érosion. Elle permet aussi de retrouver la taille normale des objets de l'image.

a)Image originale *b)filtre* 3×3 *c)filtre* 9×9

FIG. 1.13 – Le filtrage par fermeture

1.10 Conclusion

Nous avons présenté dans ce chapitre les différents techniques de filtrage, celles du filtrage linéaire permettent d'alléger le bruit impulsionnel en comparant le pixel en question avec son voisinage, ils sont donc utilisés pour la restitution des images qui ont été endommagés par des

bruits impulsionnel telle que le bruit *Salt & Pepper*[4].

Les filtres non linéaires sont plus robustes à la variation de la luminance ce qui les avantagent dans le domaine de la détection du mouvement. Pour cela nous allons consacrer un chapitre de ce mémoire aux techniques morphologiques.

[4]Il est dit aussi sel et poivre c'est un bruit constitué de pixels noirs (*poivre*) et des pixels blancs (*sel*)

2

Estimation et calcul de mouvement

2.1 Introduction

L'estimation du mouvement est une quantification du mouvement simple, par vecteurs de translation, mettant en œuvre des méthodes de calcul de trajectoire. Les applications de l'estimation du mouvement sont surtout la réduction de la redondance temporelle pour la compression et l'analyse de scène. Les méthodes qui reviennent principalement dans l'estimation du mouvement sont[GRA03] :

- Les méthodes basées sur l'estimation du flot optique (Méthodes différentielles) ;
- La mise en correspondance de blocs ;
- Les méthodes fréquentielles ;
- Les méthodes de multi-résolution, ...

La mesure du champ des vitesses est une étape de traitement de l'image dite de bas niveau. On lui trouve de nombreuses applications comme : l'analyse du mouvement des fluides, la

compression des séquences d'images vidéo par compensation de mouvement, ou son utilisation pour des phases de traitement des images de plus haut niveau, comme la reconstruction des scènes tridimensionnelles.

2.2 Mouvement réel, mouvement apparent et mouvement estimé

Les images représentent souvent la projection de scènes réelles 3D. C'est pourquoi le mouvement observé (ou mouvement apparent) dans une séquence temporelle d'images représente généralement la projection du mouvement 3D dans le plan image. On doit différencier [GRA03] :

- Le mouvement réel
- Le mouvement apparent sur l'image
- Le mouvement estimé

2.2.1 Champ de mouvement réel et champ de mouvement apparent

Le mouvement (ou déplacement) réel anime la scène réelle, dans un espace tridimensionnel. Ce mouvement est observé par une prise de vue (caméra). Quant au mouvement observé ou apparent sous la forme d'une séquence d'images 2D ou 3D, si l'on s'appuie seulement sur l'intensité des pixels. Dans ce cas on observe, en fait des changements de la distribution spatiale de l'intensité lumineuse. Le mouvement ainsi perçu est appelé champ de *mouvement apparent* ou *flot optique* qui est en général différent du champ réel de mouvement.

Le champ de déplacement apparent représente en général la projection (orthographie) du mouvement réel, il est appelé aussi *mouvement projeté*. Ce dernier représente une approximation du mouvement réel.

On suppose que le point P_t d'un objet réel 3D, à l'instant t, devient le point $P_{t'}$ à l'instant t'. La projection perspective des points P_t et $P_{t'}$ dans le plan image est notée p_t et $p_{t'}$, respectivement avec $I_t(x, y)$ et $I_{t'}(x, y)$ leurs intensités. Le mouvement apparent $\overrightarrow{p_t p_{t'}}$, correspond à

la projection perspective du mouvement réel.

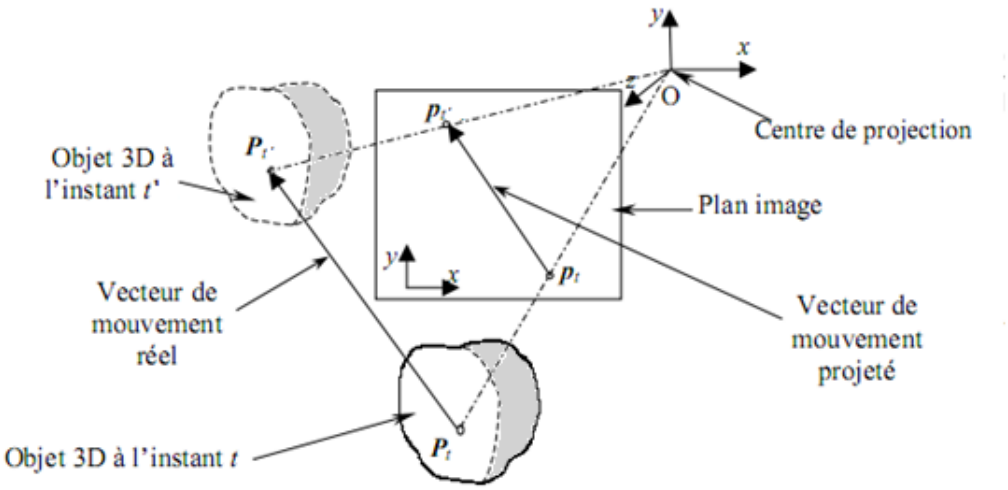


FIG. 2.1 – Illustration des mouvements réels et apparents dans un système optique de prise de vues

Dans ce cas, le mouvement projeté peut représenter la résultante du mouvement de plusieurs objets. On pourrait par exemple obtenir un mouvement nul, du fait de combinaisons inversées de deux ou plusieurs objets en mouvement. Dans le cas continu, le champ de déplacement apparent est défini pour tous les points $P(x, y, t)$ du plan image par un vecteur déplacement $d(x, y, t)$.

2.2.2 Champ de mouvement apparent et champ de mouvement estimé

Le vecteur déplacement estimé $d(p) = (d_x(p), d_y(p))$ correspond au déplacement du point $p(x, y)$ du plan image, est déterminé à partir du champ de mouvement apparent c'est-à-dire des variations locales d'intensité lumineuse $I(x, y, t)$ entre les instants t et $t + \Delta t$, où Δt est la distance en temps inter-images, dans la cas continu.

Le vecteur vitesse estimé $\overrightarrow{v} = (v_x(p), v_y(p))$ est défini comme la variation temporelle du déplacement par unité de temps $(v_x, v_y) = (\frac{dx}{dt}, \frac{dy}{dt})$. Ceci explique pourquoi une séquence temporelle d'images ne permet pas d'estimer le mouvement apparent observable dans la séquence,

et non le champ de vitesse réel. On nomme champ de déplacement, le champ de vecteurs déplacements estimés.

La différence entre le champ estimé et le mouvement apparent existe si le gradient spatial d'intensité est trop faible. Pour que le mouvement réel soit observable, il faut que la variation de niveaux de gris (respectivement de couleurs) soit suffisamment grande dans les régions ou il y a un mouvement. L'exemple classique qui met en évidence l'importance de ce facteur, est celui d'une sphère ayant une distribution uniforme d'intensité (éclairage homogène), qui effectue un mouvement de rotation autour de son axe propre, dans une scène à éclairage constant. Même si en réalité il y a mouvement, celui-ci ne peut être observé à partir d'une séquence . De plus si l'illumination de la scène varie. Un mouvement observable dans une séquence d'images d'intensité ne correspond pas toujours à un mouvement réel. La même sphère immobile soumise à une illumination variable au cours de la séquence génère un mouvement apparent (c'est-à-dire une variation d'intensité) artificiel.

2.3 Estimation du mouvement

Pour estimer le mouvement à partir du mouvement apparent, il faut admettre que l'intensité reste inchangée ou elle varie d'une façon prédictible. Cette hypothèse dite de conservation s'exprime par l'équation DFD (Displaced Frame Difference) des différences entre les images déplacées, autrement dit les images prises entre les instants t et $t + dt$.

$$DFD = I(x + dx, y + dy, t + dt) - I(x, y, t) = 0$$

L'estimation du mouvement réel à partir du mouvement apparent peut être abordée de deux manières différentes

– L'estimation des vecteurs déplacements dans le plan image $d(x, y, t) = (dx, dy)$ estimés entre les images à t et $t + \Delta t$

– L'estimation du vecteur vitesse $v(x, y, t) = (v_x, v_y)$. Les vecteurs déplacements estimés peuvent varier en espace et en temps.

On distingue deux types d'estimation : directe et inverse.

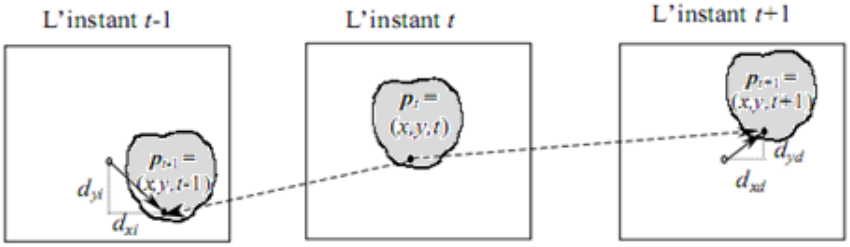


FIG. 2.2 – Estimation directe et inverse des vecteurs déplacement

Pour la figure précédente :

– Dans le cas de l'estimation directe on a noté : $d_d(x, y, t) = (d_{xd}(x, y, t), d_{yd}(x, y, t))$.

– Dans le cas de l'estimation inverse on a noté : $d_i(x, y, t) = (d_{xi}(x, y, t), d_{yi}(x, y, t))$.

2.3.1 Estimation du vecteur vitesse

Les échantillons $I(x, y, t)$ permettent de déterminer les vecteurs vitesse v. On observe que si la vitesse reste constante dans l'intervalle Δt entre deux images et si Δt est petit, alors la vitesse estimée peut être assimilée au déplacement :

$$v = \frac{d(x, y, t)}{dt}$$

Pour un mouvement accéléré, on doit élargir nos choix en prenant plus de deux images afin de mieux estimer le champ de vitesse.

En conclusion, le champ de mouvement estimé peut être caractérisé par le champ de vecteurs vitesse ou par le champ de vecteurs déplacement. L'estimation du mouvement représente donc deux approches équivalentes. L'estimation du mouvement est très sensible au bruit présent dans les images, il peut être interprété comme étant le résultat d'un mouvement dans la scène réelle. L'estimation du mouvement est un problème mal-posé qui nécessite l'ajout de contraintes.

$$\begin{cases} x = f\dfrac{X}{Z} \\ \\ y = f\dfrac{Y}{Z} \end{cases}$$

X, Y et Z sont les coordonnées d'un point de la scène

x et y sont les coordonnées du même point projeté dans le plan image
et f la distance focale de la caméra.

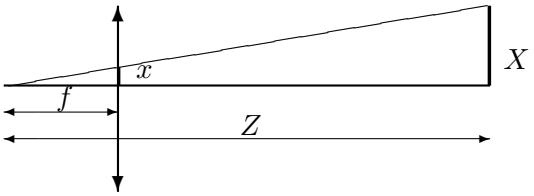


FIG. 2.3 – Projection de l'objet sur le plan d'image

2.3.2 Le calcul du mouvement

C'est l'extraction des caractéristiques physiques de l'environnement à partir d'une information fournie par l'image qui ne permet de les recouvrir que partiellement. Par exemple, on déduit la vitesse réelle d'un objet qui se déplace dans un environnement tridimensionnel, or l'information recueillie de l'image ne nous donne que deux paramètres : la vitesse horizontale v_x et la vitesse verticale v_y, ils sont appelés aussi les vitesses apparentes.

Pour vaincre cette difficulté, il nous faut des informations de plus, la distance de l'objet et la distance focale de la caméra pour trouver les vraies coordonnées de l'objet avec seulement les deux coordonnées apparentes sur notre écran. On envisage plusieurs méthodes de calcul de mouvement, chacune de ces méthodes s'appuie sur un algorithme qui lui est particulier. En général le temps réel est exigé même pour la compression dans le cas d'une transmission indirecte. Le temps de calcul est donc une préoccupation primordiale dans le traitement des séquences d'images. La figure (FIG - 2.4) représente une méthode d'estimation du mouvement, en se basant sur l'évolution de l'objet en question sur un intervalle de temps entre t et $t + \Delta t$. La matrice résultante nous nous révèle la région ayant subit un changement et ceci s'effectue en comparant les éléments de la matrice de différence avec l'image de référence.

0	252	142	13	0
0	28	23	25	0
0	78	52	42	0
0	0	0	0	0
0	0	0	0	0

-

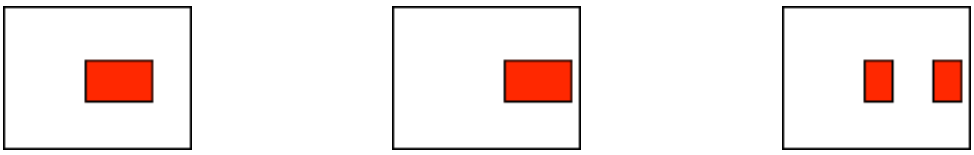
0	0	252	142	13
0	0	28	23	25
0	0	78	52	42
0	0	0	0	0
0	0	0	0	0

=

0	252	-110	-129	-13
0	28	-5	2	-25
0	78	-26	-10	-42
0	0	0	0	0
0	0	0	0	0

Changement = Mouvement

On trouve d'autres algorithmes qui ont été développés en se basant sur cette même méthode. Afin de donner des résultats plus rapidement, rappelons nous que dans ce domaine (surtout la télésurveillance), le traitement se fait en temps réel, pour cela il est inutile de comparer tout les éléments de la matrice mais seulement ceux qui constituent les éléments des contours. On admet donc que l'algorithme de détection des contours est indispensable.



à l'instant t à l'instant $t + \Delta t$ La différence

FIG. 2.4 – Calcul du mouvement entre deux images

Inconvénient : Dédoublement des objets mobiles.

2.3.3 Le champ des vitesses réelles

Admettant qu'on échantillonne 25 images/sec ($f_e = 50Hz$ pour les téléviseurs de norme PAL), donc la différence temporelle entre deux images qui se succèdent est de $\Delta t = \frac{1}{25}s = 40ms$, ensuite on fait la différence ΔX et ΔY et en divisant par Δt, on aura la vitesse instantanée suivant l'axe des x et l'axe des y dans l'intervalle de temps $[t, t + 40ms]$. Une fois qu'on a établi l'expression du flot optique on peut déduire les vitesses de nos objets si on s'intéresse à la surveillance du territoire ou bien la poursuite des cibles. $V_x = \frac{f}{Z} \cdot v_x$ et $V_y = \frac{f}{Z} \cdot v_y$, avec V_x et V_y sont les vitesses réelle de l'objet suivant l'axe des x et des y, mais on est toujours

confronté à un problème qu'on ne peut pas déterminer la vitesse normale au plan de l'image c'est-à-dire V_z .

2.3.4 Le calcul du flot optique [MAN05b]

Le problème qui se pose est la mesure du mouvement dans une séquence d'images. En général, l'évolution de l'image au cours du temps est due principalement à deux facteurs :

- des sauts entre deux séquences successives, qui sont rares et ponctuels ;
- le déplacement relatif des objets filmés et de la camera.

Le mouvement relatif des objets et de la camera est un champ de vecteurs, à trois composantes de vitesses des objets filmés dans le référentiel de la camera. Ce champ de vecteurs correspond au mouvement réel. La scène étant projetée sur le plan du film de la camera, on définit sur le plan film (ou plan focal de l'image) de la camera, un deuxième champ de vitesses qui est le champ de vitesses projeté. On note p l'opérateur de projection (qui peut être linéaire ou non). Pour chaque point a de l'image, qui est le projeté $p(a)$ d'un point réel A de vitesse V , le flot optique en x est alors le vecteur $v = \frac{dp(X)}{dt}$. L'objet de la mesure du flot optique est de l'estimer sur la base d'une séquence d'images $I(t, a(x_a, y_a))$.

La mesure du flot optique a un certain nombre d'applications possibles. Elle peut servir pour faire de la compression de séquences d'images vidéo par compensation de mouvement (prédiction d'images sur la base d'un champ de déplacement). La mesure du flot optique sert également à l'analyse de scènes : le mouvement apparent des objets d'une scène peut permettre de reconstruire une scène tridimensionnelle si on dispose d'informations supplémentaires sur la nature du mouvement réel. Ces techniques servent donc pour la construction de modèles tridimensionnels d'objets réels (acquisition tridimensionnelle) pour la réalité virtuelle, ou encore en robotique, pour construire une représentation de l'environnement d'un robot en déplacement.

Le calcul du flot optique consiste à extraire un champ de vitesses à partir d'une séquence d'images en supposant que **l'intensité (ou la couleur) est conservée** au cours du déplacement. Sous cette hypothèse, on peut établir une relation entre la vitesse apparente v (déplacement dans l'image d'indices visuels tels que des régions délimitées par des contours supposé représenter la projection du mouvement 3D des objets de la scène et/ou du mouvement de la caméra) et les variations spatio-temporelles de l'intensité.

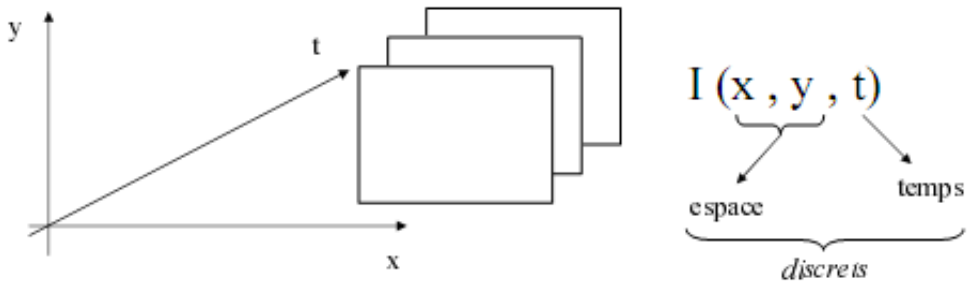


FIG. 2.5 – Les variations spatio-temporelles

Soit $I(x, y, t)$ la fonction de l'intensité lumineuse au pixel $p(x, y)$ dans l'image prise au temps t, si la luminance d'un point sur l'image ne varie pas de manière significative entre les temps t et $t + dt$ et si pendant cet intervalle de temps le point $p(x, y)$ se déplace de $d(dx, dy)$, alors En développant en série de Taylor

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \cdots$$

Et puis qu'on a mis l'hypothèse de conservation d'intensité donc

$$I(x + dx, y + dy, t + dt) = I(x, y, t)$$

Ce qui fait qu'après qu'on néglige les termes d'ordre supérieur

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0$$

Dérivant maintenant par rapport au temps

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t}\frac{dt}{dt} = 0 \qquad \Rightarrow \qquad \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0$$

Le gradient spatial est donné par $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$

Le flot optique est donné par : $v = (v_x, v_y)$ puisque il traduit les vitesses du mouvement dans l'image. Donc l'équation précédente se résume : $\nabla(I) \bullet v + \frac{\partial I}{\partial t} = 0$

$\bullet$: est le produit scalaire, Cette équation s'appelle l'équation de contraintes du mouvement.

2.4 Méthodes fréquentielles

2.4.1 Introduction

Dans le cas où on a un mouvement de translation des objet de la scène, l'image de cette translation dans le domaine fréquentiel est un déphasage, d'où ces techniques sont fondées par l'équivalence translation/déphasage de la transformée de Fourier. Donc si on considère que l'image $I(x, y)$ est une fonction bidimensionnelle discrète d'une largeur w et hauteur h, alors sa transformée de Fourier sera de la forme

$$F(u, v) = \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} I(x, y) e^{-2j\pi(ux+vy)/wh}$$

Inversement, l'image peut être décomposée en un ensemble de sinusoïdes complexes, par la transformée de Fourier inverse

$$I(x, y) = \frac{1}{wh} \sum_{u=0}^{w-1} \sum_{v=0}^{h-1} F(u, v) e^{2j\pi(ux+vy)/wh}$$

L'une des propriétés de la transformée de Fourier est la *translation*, donc pour l'image $I(x, y)$, si on se translate de $(\delta x, \delta y)$ alors sa transformée aura un déphasage de $2\pi(u\delta x + v\delta y)/wh$ d'où

$$I(x, y) \xrightarrow{\mathsf{TF}} F(u, v)$$

$$I(x + \delta x, y + \delta y) \xrightarrow{\mathsf{TF}} G(u, v) = F(u, v) e^{2j\pi(u\delta x+v\delta y)/wh}$$

avec $G(u, v)$ est la transformée de Fourier translatée de $I(x, y)$, et comme

$G(u,v) = \|G(u,v)\|e^{j\phi_G(u,v)}$ et $F(u,v) = \|F(u,v)\|e^{j\phi_F(u,v)}$ alors $\Delta\phi(u,v) = 2\pi(u\delta x + v\delta y)/wh$. Il est donc suffisant de calculer le déphasage pour deux couples (u,v) afin de trouver la translation $(\delta x, \delta y)$, par contre cette méthode est très sensible au bruit ainsi que le changement d'illumination, qui induisent des variations au niveau de basses fréquences.

2.4.2 Corrélation de phase[MAN05b]

Les méthodes de corrélation de phase estiment le déplacement relatif entre deux images consécutives en utilisant le spectre croisé normalisé (SPCN) calculé au moyen de la transformée de Fourier. Le résultat est une impulsion de *Dirac* dont la position correspond au vecteur de translation. Supposons que I_t et I_{t+1} sont deux images correspondent aux instants t et $t+1$ respectivement alors la fonction d'inter-corrélation entre ces deux images est

$$c_{t,t+1}(x,y) = I_{t+1}(x+\delta x, y+\delta y) \star I_t(-x,-y)$$
$$c_{t,t+1}(x,y) \xrightarrow{\mathsf{TF}} C_{t,t+1}(u,v)$$
$$C_{t,t+1}(u,v) = F_{t+1}(u,v) \cdot F_t^*(u,v)$$

$C_{t,t+1}(u,v)$: le spectre croisé (d'inter-corrélation).

$\star$: le produit de convolution

$$\chi = \frac{F_{t+1}(u,v) \cdot F_t^*(u,v)}{\|F_{t+1}(u,v) \cdot F_t^*(u,v)\|} = e^{-2j\pi(u\delta x + v\delta y)/wh}$$

χ : le spectre croisé normalisé (SPCN)

$$\chi \xrightarrow{\mathsf{TF}^{-1}} \delta(x-\delta x, y-\delta y)$$

$\delta(x-\delta x, y-\delta y)$: la fonction *Dirac* centrée au point $(\delta x, \delta y)$, appelée fonction de corrélation de phase.

La méthode de corrélation de phase consiste à :

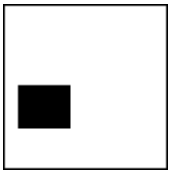
– Calculer les TF de I_t et I_{t+1}

– Calculer le SPCN des deux transformées précédentes

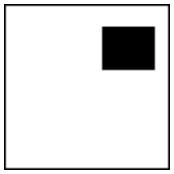
– Calculer le TF^{-1} du SPCN ou la fonction de corrélation de phase

– Rechercher le maximum de la fonction de corrélation de phase

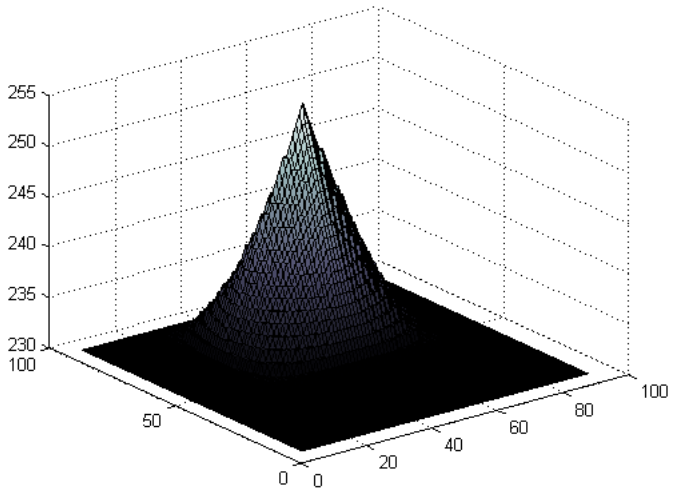
Dans le cas idéal on a la présence d'un seul pic indiquant le déplacement relatif entre les deux images, mais en pratique la fonction de corrélation peut présenter plusieurs pics à cause de l'utilisation de la $TFD^{(1)}$, de bruit ou de la présence de plusieurs objets en mouvement.



a) L'objet à l'instant t

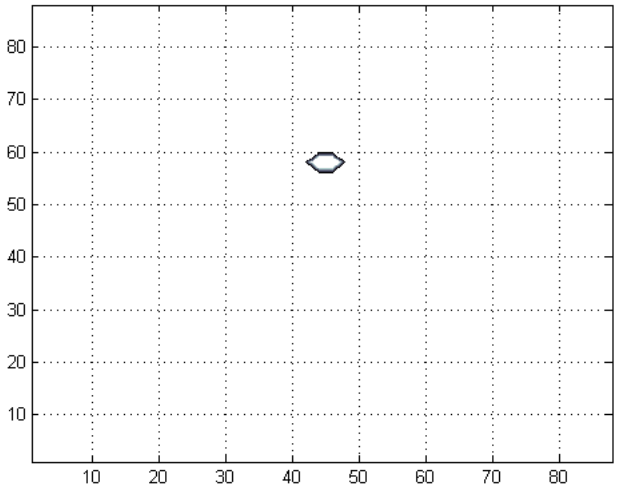


b) L'objet à l'instant $t+1$



c) La fonction de corrélation de phase

[1] TFD : TF Discrète



d) Les coordonnées du vecteur de déplacement

FIG. 2.6 – La méthode de corrélation de phase

Parmi les avantages de cette méthode on peut citer :

– *L'insensibilité aux variations de l'illumination* de la scène car la fonction de corrélation de phase ne dépend que du déplacement entre les deux images.
– *La possibilité de traiter plusieurs objets en mouvement* même s'ils ont des vitesses différentes à cause de la présence de plusieurs pics dans la fonction de corrélation de phase.
– *La rapidité de calcul* à cause de l'algorithme de calcul FFT[(2)] qui calcule la TFD plus rapidement.

En pratique cette méthode est rarement utilisée à cause de :

– *Les effets de bord* : pour obtenir une impulsion parfaite, le déplacement doit être cyclique, ce qui n'est pas le cas toujours, car les objets qui disparaissent d'un bord du bloc ne réapparaissent pas sur l'autre bord, ce qui entrainera la présence d'un pic non souhaité conduisant à une fausse estimation du mouvement.

[2] FFT : Fast Fourier Transform (TFrapide)

– *Déformations du spectre* dues aux valeurs non entières des vecteurs de mouvement. Pour obtenir un pic idéal, les composantes du vecteur déplacement doivent correspondre à un multiple de la fréquence fondamentale, si ce n'est pas le cas alors des pics apparaitront et déformeront le spectre.

– *L'amplitude maximale du déplacement* qui peut être estimée, car la TFD est périodique, et d'une période de $B_x \times B_y$ tel que B_x, B_y sont les dimensions d'un bloc, alors le déplacement estimé doit être reformulé comme suit

$$\hat{d}_i = \begin{cases} d_i & , |d_i| \leq E(d_{max})/2 \\ d_i - d_{max} & , ailleurs \end{cases}$$

$E(x)$: est la partie entière de x

afin de permettre des déplacements négatifs. Ainsi, si le déplacement d_{max} est pair, l'intervalle d'estimation est $[-d_{max}/2+1, d_{max}/2]$, alors pour estimer un déplacement de $d_{max}/2$ il nous faut un bloc de $d_{max} \times d_{max}$ pixels, ce qui est très grand.

2.4.3 Transformée de Fourier-Mellin [MAN05b]

La transformée de Fourier-Mellin (TFM) permet d'estimer le mouvement relatif, en déterminant les paramètres (α, σ) qui correspondent respectivement à une (*rotation et homothétie*), comme un vecteur de translation d'une manière analogue au cas de la méthode de corrélation de phase, grâce à la représentation *log-polaire* de l'espace fréquentiel d'où : $(u, v) \longrightarrow (\theta, \log \rho)$

Soit g l'image transformée de f par une rotation d'angle α, une homothétie de rapport σ, et d'une translation de vecteur (x_0, y_0), alors elle s'écrit sous la forme suivante :

$$g(x, y) = f(\sigma(cos\alpha \cdot x + sin\alpha \cdot y) - x_0, \sigma(-sin\alpha \cdot x + cos\alpha \cdot y) - y_0)$$

$$g(x, y) \xrightarrow{\text{TF}} G(u, v)$$

ce qui donne

$$\|G(u, v)\| = \frac{1}{\sigma^2}\|F(\frac{1}{\sigma}(u\cos\alpha + v\sin\alpha), \frac{1}{\sigma}(-u\sin\alpha + v\cos\alpha))\|$$

On remarque que l'amplitude dans l'espace fréquentiel ne dépend pas de la translation (x_0, y_0), par contre elle a subi une rotation de α ainsi qu'une modification d'échelle de $\frac{1}{\sigma}$.

L'étape suivante consiste à changer les coordonnées fréquentielles (u, v) par des coordonnées polaires (θ, ρ), d'où les deux fonction F_p et G_p s'écrivent sous la forme suivante

$$F_p(\theta, \rho) = \|F(\rho\cos\theta, \rho\sin\theta)\|; \qquad 0 \leq \theta \leq 2\pi, 0 \leq \rho < \infty$$

$$G_p(\theta, \rho) = \|G(\rho\cos\theta, \rho\sin\theta)\|; \qquad 0 \leq \theta \leq 2\pi, 0 \leq \rho < \infty$$

On remplaçant (u, v) par $(\rho\cos\theta, \rho\sin\theta)$ on trouvera :

$$G_p(\theta, \rho) = \frac{1}{\sigma^2} F_p(\theta - \alpha, \frac{\rho}{\sigma})$$

Enfin, en passant la coordonnée radiale au logarithme.

$$\begin{array}{ll} r = log\rho & F_{lp}(\theta, r) = F_p(\theta, \rho) \\ s = log\sigma & G_{lp}(\theta, r) = G_p(\theta, \rho) \end{array}$$

On obtient finalement :

$$\boxed{G_{lp}(\theta, r) = \frac{1}{\sigma^2} F_{lp}(\theta - \alpha, r - s)}$$

D'après l'équation précédente, on peut bien envisager la similitude dans l'espace fréquentiel qui peut être traduit par une *translation* dans l'espace *log-polaire*

2.5 Méthodes différentielles

2.5.1 Introduction

Les méthodes différentielles sont des méthodes basées sur le gradient spatial et temporel de l'intensité lumineuse, ces gradients sont approchés dans le cas discret par des différences finies. On peut remarquer bien que l'ECM[3] ne suffit pas pour déterminer le champ de vitesses (*flot optique*), car elle contient deux inconnues (v_x, v_y), La composante normale de v est orientée

[3] L'équation de contraintes du mouvement

selon la direction du gradient spatial de l'intensité lumineuse ($\frac{\nabla I}{\|\nabla I\|}$), par contre la composante tangentielle n'est pas accessible, car elle disparaisse à cause du produit scalaire.

$$\begin{cases} \|\nabla I\| \bullet \|v_\perp\| \cos(0) + \dfrac{\partial I}{\partial t} = 0 & \|v_\perp\| = \dfrac{\dfrac{\partial I}{\partial t}}{\|\nabla I\|} \\ \|\nabla I\| \bullet \|v_\tau\| \cos(\dfrac{\pi}{2}) + \dfrac{\partial I}{\partial t} = 0 & \|v_\tau\| = ? \end{cases}$$

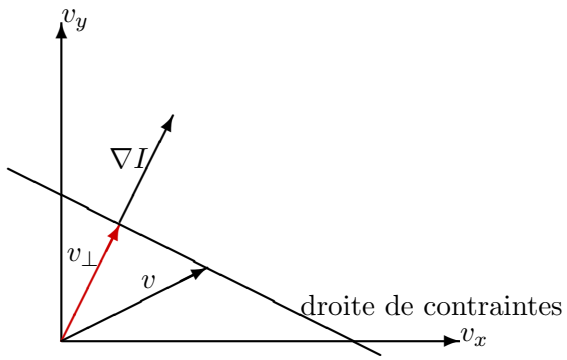


FIG. 2.7 – Illustration géométrique de l'équation et de la droite de contrainte

2.5.2 Méthode de Horn et Schunck[HS81]

La méthode de *Horn* et *Schunck* n'est pas limité à la translation, mais c'est une méthode différentielle itérative estimant les petits déplacements, qui est basée sur le développement en série de *Taylor* de la DFD, des gradients spatiaux et temporaux. Son but est de trouver un champ de vecteur satisfaisant pour résoudre le problème de l'ECM.[GRA03]

Soit :

$$\xi_b = \frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t}$$

ξ_b : l'erreur par rapport à l'équation de contrainte du mouvement (ECM). On voit bien que quand cette erreur est nulle alors l'équation DFD est satisfaite, d'où cette méthode cherche à minimiser le carré de ξ_b. Une autre contrainte supplémentaire de lissage qui suppose que tous

les points voisins ont un mouvement semblable :

$$\xi_c^2 = \|\nabla v_x\|^2 + \|\nabla v_y\|^2 = \left(\frac{\partial v_x}{\partial x}\right)^2 + \left(\frac{\partial v_x}{\partial y}\right)^2 + \left(\frac{\partial v_y}{\partial x}\right)^2 + \left(\frac{\partial v_y}{\partial y}\right)^2$$

La méthode de *Horn* et *Schunck* consiste à minimiser ξ tel que :

$$\xi^2 = \iint (\alpha \xi_c^2 + \xi_b^2) dx dy$$

α : coefficient de pondération entre les termes de l'ECM et ceux de la régulation.

Pour minimiser ξ^2 on utilise les équations d'*Euler-Lagrange*, on utilise ainsi l'approximation du Laplacien $\nabla^2 f = f - \bar{f}$, tel que $\bar{f}$ est la moyenne de f dans un certain voisinage, on obtient alors :

$$\left(\frac{\partial I}{\partial x}\right)^2 \cdot v_x + \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial y} \cdot v_y = \alpha \cdot (\bar{v}_x - v_x) - \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial t}$$

$$\left(\frac{\partial I}{\partial y}\right)^2 \cdot v_y + \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial y} \cdot v_x = \alpha \cdot (\bar{v}_y - v_y) - \frac{\partial I}{\partial y} \cdot \frac{\partial I}{\partial t}$$

Si on résoud le système d'équations précédent on obtiendra :

$$v_x = \frac{\left(\alpha + \left(\frac{\partial I}{\partial y}\right)^2\right) \cdot \bar{v}_x - \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial y} \cdot \bar{v}_y - \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial t}}{\alpha + \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

$$v_y = \frac{\left(\alpha + \left(\frac{\partial I}{\partial x}\right)^2\right) \cdot \bar{v}_y - \frac{\partial I}{\partial x} \cdot \frac{\partial I}{\partial y} \cdot \bar{v}_x - \frac{\partial I}{\partial y} \cdot \frac{\partial I}{\partial t}}{\alpha + \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Reformulant les deux équation précédentes

$$\left(\alpha + \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)(v_x - \bar{v}_x) = -\frac{\partial I}{\partial x}\left(\frac{\partial I}{\partial x}\bar{v}_x + \frac{\partial I}{\partial y}\bar{v}_y + \frac{\partial I}{\partial t}\right)$$

$$\left(\alpha + \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)(v_y - \bar{v}_y) = -\frac{\partial I}{\partial y}\left(\frac{\partial I}{\partial x}\bar{v}_x + \frac{\partial I}{\partial y}\bar{v}_y + \frac{\partial I}{\partial t}\right)$$

Maintenant qu'on a une paire d'équations pour chaque point de l'image, ça ne sera pas difficile à résoudre ces deux équations simultanément, par une des méthodes standards. La matrice correspondante est vaste et très large, tant que le nombre de ligne et de colonnes et le double de ceux de l'image. Les méthodes itératives comme celle de *Gauss-Seidel* s'imposent. Donc on peut calculer la nouvelle vitesse estimée (v_x^{k+1}, v_y^{k+1}) à partir des dérivées estimées ainsi qu'à partir de la valeur moyenne de la vitesse estimée précédente (v_x^k, v_y^k) en utilisant :

$$v_x^{k+1} = \bar{v}_x^k - \frac{\frac{\partial I}{\partial x}\left(\frac{\partial I}{\partial x}\bar{v}_x^k + \frac{\partial I}{\partial y}\bar{v}_y^k + \frac{\partial I}{\partial t}\right)}{\alpha + \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

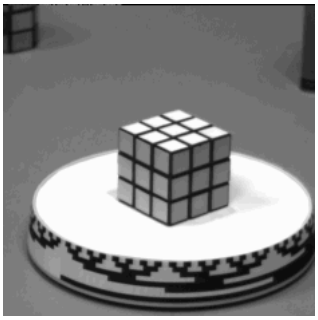
$$v_y^{k+1} = \bar{v}_y^k - \frac{\frac{\partial I}{\partial y}\left(\frac{\partial I}{\partial x}\bar{v}_x^k + \frac{\partial I}{\partial y}\bar{v}_y^k + \frac{\partial I}{\partial t}\right)}{\alpha + \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

2.6 Méthodes de corrélation de blocs

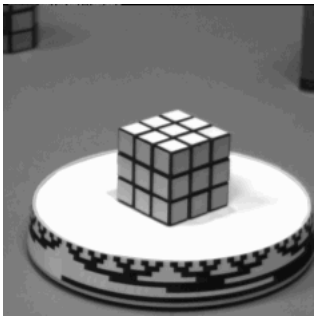
2.6.1 Introduction

Parmi les méthodes les plus utilisées dans le domaine de l'estimation du mouvement, les méthodes de corrélation de blocs BM[4], elles sont adoptées par diverses normes visuelles de codage (MPEG 1/2 et H.261/262/263), des applications telles que la stéréoscopie, surveillance de sites et trafics routiers ; grâce à leurs simplicité d'implémentation *software* et *hardware*. Ces méthodes basent sur la division de l'image courante en un ensemble de petits blocs rectangulaires, en supposant que les pixels de chacun des blocs ont le même mouvement de translation. D'où pour chaque bloc un vecteur de mouvement est obtenu grâce au calcul des coordonnées déplacées d'un bloc correspond dans l'image de référence.

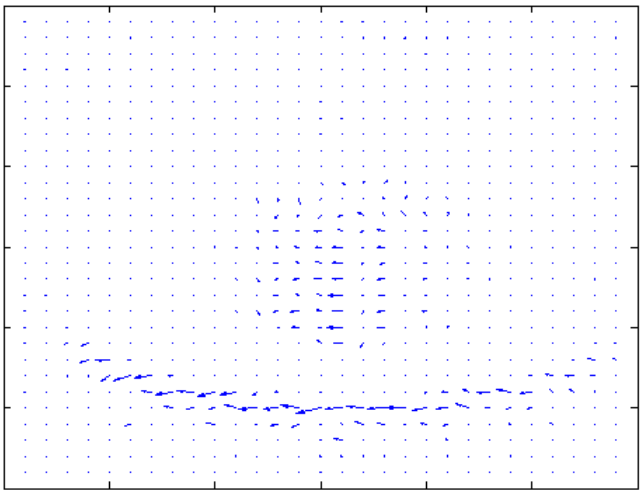
[4] BM : block matching



a) L'objet à l'instant t



b) L'objet à l'instant $t+1$



c) La représentation du *flot optique*

FIG. 2.8 – La méthode de Horn et Schunck pour le calcul du flot optique

2.6.2 Critères de corrélation de blocs

Les mouvements relatifs des objets de la scène induisent des déplacement au cours du temps des motifs correspondant à leur projection dans l'image. La fréquence d'échantillonnage temporel à la cadence vidéo limite les changements brutaux et génère des mouvements apparents d'amplitude réduite. Cela induit une redondance qui justifie pleinement le développement d'opérateurs d'estimation de mouvement ou de déplacement. Ces méthodes permettent une estimation des mouvements de translation. Les images sont divisées en un ensemble de blocs de même taille, chacun entre eux est associé à un vecteur de déplacement obtenu par une mesure

d'une certaine corrélation.[SS01]

On considère que l'image I_t est divisée en un ensemble de blocs de taille $N \times N$ pixels, on cherche ensuite un bloc de taille $N \times N$ dans l'image I_{t-1} qui minimise un certain critère de distance entre les deux blocs. Cette recherche est effectuée à l'intérieur d'une fenêtre de taille $(2d+N) \times (2d+N)$, centrée sur la position du bloc courant de l'image I_t. Cette fenêtre limite la recherche à une distance de $\pm d$ autour du bloc courant de l'image I_{t+1}. L'algorithme consiste à calculer à chaque point $(x, y) \in [-d, +d]$ la distance :

$$EAM(x,y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |I_t(i,j) - I_{t-1}(i+x, j+y)|$$

EAM : l'erreur absolue moyenne

On cherche ensuite le minimum

$$M = min_{(x,y)}(EAM(x,y))$$

La valeur de $EAM(x,y)$ doit être calculé pour tous les blocs de l'image I_{t-1} qui sont à l'intérieur de la fenêtre de recherche. La valeur de M est le minimum de toutes les EAM de cette fenêtre. On appelle le *vecteur de mouvement* la distance minimale entre le bloc courant de l'image I_t et le bloc de I_{t-1} résultant de M.

Le critère de distance EAM est le plus souvent employé car il est moins couteux en calcul que celui basé sur l'erreur quadratique moyenne (EQM) qui est définit par :

$$EQM(x,y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I_t(i,j) - I_{t-1}(i+x, j+y))^2$$

On a ainsi un troisième critère de distance appelé *classification de la différence entre pixels* (PDC)[5], celui-ci est rarement utilisé, il classifie les pixels en deux catégories

– les pixels corrélés

[5] PDC : Pixel Difference Classification

– les pixels qui diffèrent

On calcule cet critère de la manière suivante :

$$PCD(x,y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} T(i,j,x,y)$$

avec

$$T(i,j,x,y) = \begin{cases} 1 & si \quad |I_t(i,j) - I_{t-1}(i+x, j+y)| \leq \tau \\ 0 & ailleurs \end{cases}$$

τ : un seuil de classification prédéfini

D'après l'équation de $T(i,j,x,y)$ on peut dire que ce critère cherche le plus grand nombre de pixels qui se ressemblent entre les deux bloc, plus le nombre de pixels semblables est grand, plus le bloc a la chance d'être choisi, et plus la valeur de T est proche de $(N-1)^2$ plus la corrélation est meilleure. Ce qu'on remarque aussi est que ce critère est plus complexe que l'EAM car une opération supplémentaire de comparaison est nécessaire pour chaque pixel.

La façon la plus simple pour rechercher des blocs dans la fenêtre est l'approche exhaustive (*full-search*), qui examine tous les blocs possibles. Mais afin d'optimiser la recherche dans la fenêtre tout en réduisant le volume de calcul, plusieurs familles d'algorithmes sont développés, ces algorithmes sont appelés les approches non-exhaustives, où ils diminuent le nombre de blocs candidats[6] ou le nombre de pixels par bloc.

2.6.3 Approche exhaustive

L'approche exhaustive est la plus simple et régulière en calcul, car à la recherche on teste tous les blocs candidats, par contre elle est la plus coûteuse en volume de calcul, celui-ci dépend de la taille de la fenêtre de recherche. Le nombre de blocs candidats pour une distance de recherche de d est de $(2d+1)^2$.

[6] Les blocs candidats sont ceux de l'image I_{t-1}

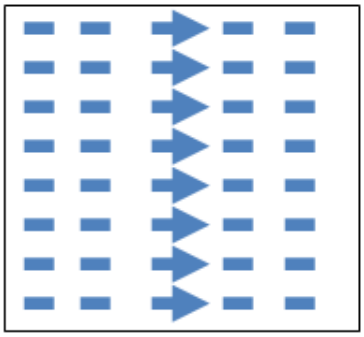


FIG. 2.9 – L'approche exhaustive

2.6.4 Approches non-exhaustives

Ces approches consiste en une exploration initiale du voisinage direct de la fenêtre de recherche, puis une exploration sélective dans une direction de ressemblance croissante. Il existe de nombreux algorithmes de recherche non-exhaustive, parmi les plus utilisés :

- la recherche en n pas
- la recherche 2D-logarithmique
- la recherche orthogonale OSA (*Orthogonal Search Algorithm*)

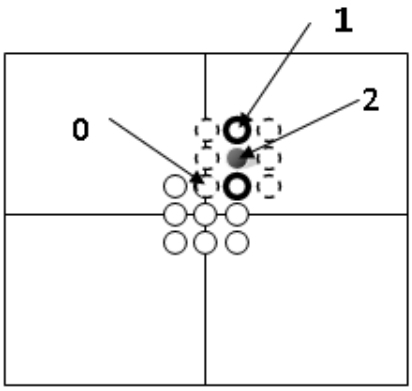


FIG. 2.10 – L'approche non-exhaustive

L'algorithme à n-pas

Cet algorithme est une approche dichotomique qui cherche le bloc candidat optimum en basant sur un raffinement successif. Soit d_{max} le déplacement maximal (*la taille de la fenêtre de recherche*) et O le pixel central de la fenêtre. A la première étape, le critère de ressemblance est évalué en 9 blocs : le bloc centré en O ainsi que ces 8 voisins distants de $\lceil d_{max}/2 \rceil$ pixels, tel que $\lceil x \rceil$ est l'arrondi de x à la valeur entière la plus supérieure. A chaque itération, on évalue le critère de ressemblance et l'optimum devient le centre de recherche pour l'itération suivante. La distance entre le bloc central et ces 8 voisinage décroît à chaque itération suivant la loi

$$d_{(k)max} = \lceil d_{(k-1)max}/2 \rceil$$

tel que $d_{(k)max}$, $d_{(k-1)max}$ sont respectivement la distance de l'itération courante (k) et précédente $(k-1)$. L'estimation du mouvement est obtenue lorsque la distance $d_{(k)max} = 1$, d'où le nombre de distances entre blocs à calculer afin d'obtenir le bloc optimal est $m = 8\lceil \log_2(d_{max}) \rceil$, ainsi le nombre de pas n est égal à $\lceil \log_2(d_{max}) \rceil$

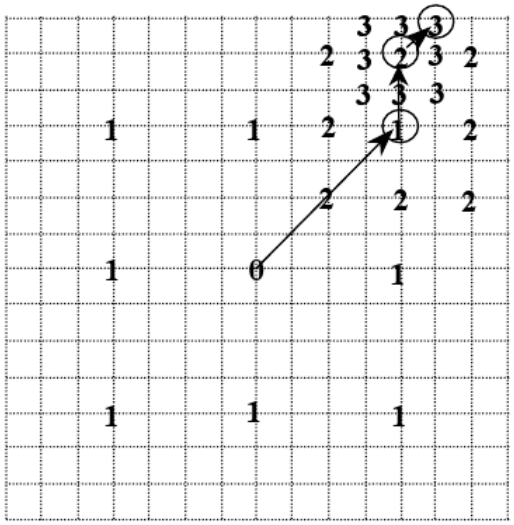


FIG. 2.11 – Approche non-exhaustive (L'algorithme à 3-pas)

L'algorithme 2D-logarithmique

La stratégie de cet algorithme de recherche est en croix (+) à chaque itération. On initialisant le pas à $\lceil d_{max}/4 \rceil$, ce dernier ne se réduit à la moitié que quand le bloc optimal de l'étape courante se trouve au centre du croix ou au bord de la fenêtre. Quand le pas atteint 1, les 8 blocs immédiatement voisins du pixel central sont testés. La figure FIG - 2.12 nous montre les deux cas de réduction de pas.

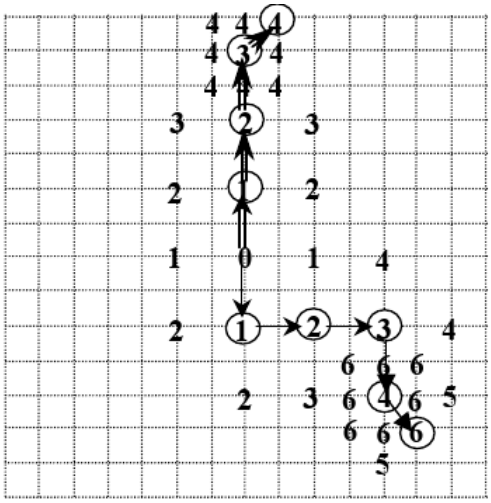


FIG. 2.12 – Approche non-exhaustive (2D-logarithmique)

L'algorithme orthogonal OSA

Cet algorithme hybride est implémenté des deux algorithmes :

– l'algorithme à 3-pas

– l'algorithme 2D-logarithmique [SS01]

C'est une suite de comparaisons horizontale et verticale, avec une décroissance logarithmique du pas. La distance initiale est de $\lceil d_{max}/2 \rceil$. Chaque itération est divisée en deux étape comparaison horizontale suivie d'une autre verticale, à chaque étape on cherche le bloc optimum parmi trois blocs : Le bloc central et les deux autres voisins soit horizontalement ou vertica-

lement. Après chaque étape le pas est diminué en moitié. Ce processus s'arrête quand le pas sera égal à 1.

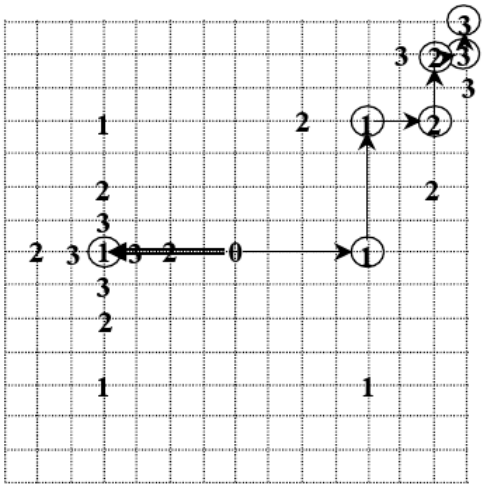


FIG. 2.13 – Approche non-exhaustive (recherche orthogonale)

2.7 Conclusion

Nous avons présenté dans ce chapitre, plusieurs méthodes d'estimation du mouvement, dont chacune d'elles présentent des avantages et des inconvénients, nous ne pouvons pas donc privilégier une méthode que par son domaine d'utilisation. Les méthodes fréquentielles sont très robustes envers la variation de luminance puisque elles comparent deux images et ne nous retournent qu'une impulsion indiquant les paramètres du déplacement effectué (les coordonnées de translation pour la corrélation de phase, l'angle de rotation α et le facteur d'homothétie σ pour la TFM). Les méthodes fréquentielles ont l'avantage aussi du traitement de plusieurs objets en parallèle. Tandis que les méthodes différentielles (Horn et Schunck) permettent d'estimer n'importe quel mouvement à condition qu'il soit un mouvement infinitésimal, c'est pour cela qu'on doit échantillonner avec plus de précision et donc cette méthode est très coûteuse en terme d'occupation de mémoire. Les méthodes de corrélation de blocs s'appuient sur la recherche des blocs qui se ressemblent, elle regroupe deux méthodes : les méthodes exhaustives

qui sont plus raffinées, mais si on est limité par le temps on fait appel aux méthodes non-exhaustives qui nous donnent des résultats de moindre qualité mais elles sont beaucoup plus rapides.

3

Les méthodes morphologiques

3.1 Introduction

La morphologie mathématique est une théorie essentiellement non linéaire, c'est un ensemble de méthodes d'analyse d'images. Son utilisation en détection et en estimation du mouvement est relativement récente, dont le but est l'étude des objets en fonction de leur forme, de leur taille, des relations avec leur voisinage, de leur texture, et de leurs niveaux de gris ou de leur couleur. Par les transformations qu'elle propose, elle se situe à différents niveaux du traitement d'images (filtrage, segmentation, mesures, analyse de texture) et fournit ainsi des outils pour la reconnaissance des formes. La morphologie mathématique, développée à l'origine pour l'étude des matériaux poreux, trouve maintenant ses applications dans de nombreux domaines du traitement d'images, aussi bien 2D que 3D, en biologie et cytologie quantitative, en imagerie médicale, en imagerie aérienne et satellitaire, en robotique et vision par ordinateur, dans les études sur les documents et les oeuvres d'art.

Le principe de base de la morphologie mathématique est de comparer l'image à analyser par rapport à un ensemble de géométrie connue appelé élément structurant que l'on déplace de façon à ce que son origine passe par toutes les positions de l'image, pour mettre en évidence certaines caractéristiques de l'image. En suivant ce principe, il est alors possible de :

- Rechercher la plus grande valeur, ou la plus petite, dans le domaine de l'image défini momentanément par la présence de l'élément structurant et l'affecter au pixel de l'image sur lequel le centre de l'élément structurant est positionné. On définit ainsi respectivement l'opérateur de dilatation ou celui d'érosion.
- Rechercher dans l'image, une configuration géométrique de pixels particulière correspondante à celle de l'élément structurant utilisé, ceci définit les opérateurs d'épaississement et d'amincissement. A la différence de ceux utilisés pour une dilatation ou pour une érosion, les éléments structurants des épaississements ou des amincissements sont définis aux rotations près, ceci signifie que pour chaque pixel de l'image, on fait tourner le centre de l'élément structurant dans toutes les directions de la trame et on raisonne, pour chacune d'elle, de la même façon. Notons que l'érosion et la dilatation sont respectivement un cas particulier de l'amincissement et de l'épaississement.

La différentiation trame à trame se fait par l'utilisation d'éléments structurants spatio-temporels. Elle permet de calculer des filtres spatio-temporels, et d'intégrer des changements temporels par accumulation. 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 Z^3 avec deux dimensions spatiales x, y et une dimension temporelle t, un élément structurant spatio-temporel B est défini comme un voisinage de l'origine dans Z^3. Considérons une séquence d'image $I_t(x, y)$ ou t est un index de temps et (x, y) un index de l'espace bidimensionnel, les filtres morphologiques temporels sont définis en utilisant un élément structurant temporel $\tau = [t1, t2]$ avec $t1$ et $t2$ deux instants tels que $t1 < t2$.

On définit dans ce qui suit, les principaux types de morphologies.

3.2 Morphologie temporelle

3.2.1 Erosion temporelle ϵ

L'érosion temporelle $\epsilon_\tau(x, y, t)$ se définit donc par

$$\epsilon_\tau(x, y, t) = min(I_{t+z}(x, y)) \qquad z \in \tau$$

a) L'images originale à t

b) L'images originale à $t + 4$

c) Erosion temporelle à t pour $\tau = [0, 2]$

d) Erosion temporelle à $t+4$ pour $\tau = [0, 2]$

FIG. 3.1 – L'érosion temporelle

3.2.2 Dilatation temporelle δ

La dilatation temporelle $\delta_\tau(x, y, t)$ se définit donc par

$$\delta_\tau(x, y, t) = max(I_{t+z}(x, y)) \qquad z \in \tau$$

a) Dilatation temporelle à t pour $\tau = [0, 2]$

b) Dilatation temporelle à $t + 4$ pour $\tau = [0, 2]$

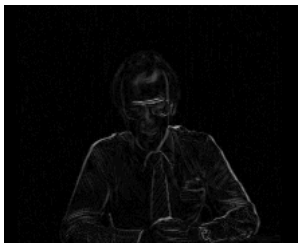
FIG. 3.2 – La dilatation temporelle

3.2.3 Gradient morphologique temporel γ

Quant au gradient γ_τ, il se définit par

$$\gamma_\tau(x, y, t) = \delta_\tau(x, y, t) - \epsilon_\tau(x, y, t)$$

τ représente un intervalle temporel d'interaction, il peut être causal, anti-causal ou les deux en même temps. On envisage de nombreuses autres méthodes de détection du mouvement qui ont été élaborées et 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.



a) Gradient temporel à t pour $\tau = [0, 2]$

b) Gradient temporel à $t + 4$ pour $\tau = [0, 2]$

FIG. 3.3 – La gradient temporel

3.3 La moyenne récursive

Soit I_t l'image acquise à l'instant t, l'expression du fond correspond M_t s'obtient à l'aide de la formule suivante

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

Le calcul est particulièrement plus rapide lorsque α est une puissance négative de 2, et correspond alors à un simple décalage de bits à droite pour notre implémentation FPGA.

- Lorsque α vaut 1, $M_t = I_t$, l'image acquise égale le fond courant.
- Lorsque α vaut 0, $M_t = M_{t-1} = M_0$, le fond reste égale à son prédécesseur.

La technique du fond récursif revient dans ce cas à une technique par différence avec une image dite de référence qui ne serait jamais remise à jour. En faisant varier α 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 changements de la scène(remise à jour du fond).

3.3.1 L'algorithme

On initialise pour chaque pixel de coordonnées (x, y)

$$M_0(x, y) = I_0(x, y)$$

Et à chaque instant t, on a aussi

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

$$D_t(x, t) = \delta_t(|M_t(x, y) - I_t(x, y))|)$$

Et la simple différence Δ se donne par la formule suivante

$$\Delta_t(x, y) = |I_t(x, y) - M_t(x, y)|$$

Mais malheureusement les filtres morphologiques récursifs présentent des inconvénients parmi lesquels, on cite :

– Ils impliquent l'utilisation d'un buffer[1], d'une taille très prohibitive donc une occupation excessive de l'espace mémoire.
– Sensibilités aux variations brusques comme un bruit impulsionnel (brusque changement de luminance), une fluctuation des capteurs.


a) Images originale à $t+6$


b) M_{t+6} pour $\alpha = 0.25$


c) Δ pour $\alpha = 0.25$

FIG. 3.4 – La moyenne récursive

3.4 La morphologie oublieuse temporelle

Après l'avènement des filtres morphologiques oublieux, leur utilisation semble inévitable puisque l'objet en mouvement est mieux segmenté et sa position actuelle est mieux définie que les opérateurs classiques. Il s'avère surtout que l'objet en mouvement dans l'image du gradient morphologique oublieux temporel est mieux visible que dans l'image du gradient morphologique temporel classique, et que les zones homogènes sans bruit sont mieux filtrées.

Les filtres morphologiques oublieux sont des filtres hybrides, maximum linéaire, et minimum linéaire. Ils combinent en effet le calcul des minima et maxima avec un terme d'oubli α.

3.4.1 L'algorithme

L'initialisation démarre d'un point commun, où la dilatation morphologique oublieuse M_0, l'érosion morphologique oublieuse m_0 sont égales à l'image de référence I_0.

$$M_0(x,y) = m_0(x,y) = I_0(x,y) \tag{3.1}$$

[1]buffer : la mémoire permettant le stockage temporaire des données(mémoire tampon)

Ensuite à chaque instant t la dilatation morphologique oublieuse est la somme de l'image courante, et la maximale entre l'image courante et la dilatation précédente multipliés par les facteurs α et 1-α respectivement comme le montre la formule ci-dessous, la dilatation oublieuse a donc le même effet que la moyenne récursive seulement cette fois-ci le facteur ajouté dépend de la comparaison entre l'image courante et la dilatation oublieuse précédente.

$$M_t(x,y) = \alpha I_t(x,y) + (1-\alpha) max(I_t(x,y), M_{t-1}(x)) \tag{3.2}$$

De même pour l'érosion oublieuse, on obtient la formule suivante :

$$m_t(x,y) = \alpha I_t(x,y) + (1-\alpha) min(I_t(x,y), m_{t-1}(x,y)) \tag{3.3}$$

On remarque bien que la condition initiale reste valable pour toute trame de facteur $\alpha = 1$

$$M_t(x,y) = m_t(x,y) = I_t(x,y) \tag{3.4}$$

Et si $\alpha = 0$

$$M_t(x,y) = (1-\alpha) max(I_t(x,y), M_{t-1}(x)) \tag{3.5}$$

$$m_t(x,y) = (1-\alpha) min(I_t(x,y), m_{t-1}(x)) \tag{3.6}$$

La dilatation (respectivement l'érosion) n'est qu'une simple maximisation (respectivement minimisation) entre l'image courante et le fond sur un intervalle $\tau = [t-1, t]$. Le gradient morphologique oublieux vient donc

$$\Gamma_t(x,y) = M_t(x,y) - m_t(x,y) \tag{3.7}$$

Ou d'une manière explicite

$$\Gamma_t(x,y) = (1-\alpha)(max[I_t(x,y), M_{t-1}(x,y)] - min[I_t(x,y), m_{t-1}(x,y)]) \tag{3.8}$$

Cette technique a les avantages d'être :

• pour $\alpha = 0.5$

a) M_{t+6}

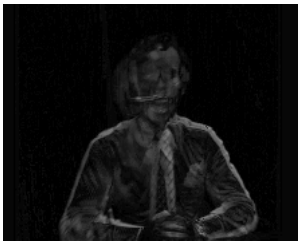
b)m_{t+6}

c)Γ_{t+6}

• pour $\alpha = 0.125$

a) M_{t+6}

b)m_{t+6}



c)Γ_{t+6}

a) M_t Dilatation oublieuse ; b) m_t Erosion oublieuse ; c) Γ_t Gradient oublieux

FIG. 3.5 – La morphologie oublieuse

- Moins sensible aux bruits impulsionnels grâce â son terme linéaire qui induit une diminution importante des poids attachés aux valeurs passées.
- Elle 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 α, ainsi de détecter les mouvements lents.
- Elle nécessite uniquement l'utilisation d'un buffer de taille 2 (images) pour calculer l'érosion et la dilatation oublieuse dans le cas d'une implémentation sur circuit.

En revanche le terme oublieux α a besoin d'être adapté à la vélocité de l'objet en mouvement. Comme il y a plusieurs objets de différentes tailles et de différentes vitesses dans la scène, il est nécessaire d'ajuster localement la valeur de α. Les filtres morphologiques oublieux ont plus d'avantages puisque ils combinent les opérations morphologiques et filtrages linéaires, Les opérations morphologiques seules sont un puissant outil de détection dans la mesure où elles

estiment l'intensité lumineuse $I(x, y)$ dans un intervalle temporel donné. Mais ils maximisent également le bruit temporel. A contrario, le filtrage linéaire seul permet de limiter les effets du bruit temporel, mais il a un pouvoir de détection moindre. La détection du mouvement à base de filtres morphologiques oublieux est particulièrement adaptée aux mouvements de faible amplitude, à des mouvements lents ou à des objets faiblement contrastés[RIC06].

3.5 L'estimation $\Sigma - \Delta$ [RIC06], [MR07]

3.5.1 Introduction

L'estimation $\Sigma - \Delta$ est une simple méthode non-linéaire de soustraction du fond, basée sur la comparaison ainsi que quelques incrémentions/décrémentions élémentaires. Elle consiste à incrémenter (*respectivement* décrémenter) la valeur estimée courante par une constante, si elle est plus petite (*respectivement* grande) que la valeur échantillonnée.

3.5.2 Le principe de l'estimation $\Sigma - \Delta$

Le signal à estimer est numérique car il s'agit de la séquence d'images I_t. On doit calculer pour chaque image I_t un fond estimé M_t, ceci prend ces valeurs avec une grande probabilité d'être dans l'intervalle $[a, b]$ tant qu'il y a autant d'indices $\tau < t$ tel que $I_t > a$, que d'indices $\tau < t$ tel que $I_t < b$, dans le cas où I_t est un signal discret aléatoire. Donc M_t est une approximation de la médiane de I_t [MR04b]. L'algorithme ci-dessous nous montre la façon de calculer le fond M_t

$sgn(x)$: est la fonction signe, qui renvoie 1 si $x > 0$, -1 si $x < 0$ et 0 quand $x = 0$

$\forall p = (x, y) \in [(1, 1), (largeur, hauteur)]$	Pour chaque pixel
$M_0(p) = I_0(p)$	Initialisation
$\forall \tau \in [0, t_{final}]$	Pour chaque trame
$\forall p = (x, y) \in [(1, 1), (largeur, hauteur)]$	Pour chaque pixel
$M_\tau(p) = M_{\tau-1}(p) + sgn(I_t(p) - M_{\tau-1}(p))$	

Mais ce filtre a une autre propriété plus intéressante, c'est que la différence entre le fond M_t et l'images I_t soit proportionnelle au taux de variation de I_t. Or cette différence Δ_t peut être mesurée précisément, et qui est un critère de vraisemblance du mouvement.

$\forall \tau \in [0, t_{final}]$	Pour chaque trame
$\forall p = (x, y) \in [(1, 1), (largeur, hauteur)]$	Pour chaque pixel
$\Delta_\tau(p) = M_\tau(p) - I_\tau(p)$	La mesure de vraisemblance

On utilise aussi ce filtre pour calculer la *variance temporelle* des pixels, celle-ci mesure leurs activité temporelle[(2)], et décide ainsi qu'un pixel est en mouvement ou non. Donc un deuxième estimateur est employé dans l'estimation $\Sigma - \Delta$. En générale on applique le filtre $\Sigma - \Delta$ à N fois la différences non nulles, car nous nous intéressons aux pixels qui ont un taux de variation significativement supérieur à leur variation temporelle. Le réglage du paramètre N n'est pas trop sensible, il est en générale entre 1 et 4 (*puissance de 2 de préférence pour des raisons d'optimisation*)[MR07], le paramètre N intervient généralement à l'accélération de la convergence de la moyenne de $\Sigma - \Delta$.

$\forall p = (x, y) \in [(1, 1), (largeur, hauteur)]$	Pour chaque pixel
$V_0(p) = \Delta_0(p)$	Initialisation
$\forall \tau \in [0, t_{final}]$	Pour chaque trame
$\forall p = (x, y) \in [(1, 1), (largeur, hauteur)]$ et $\Delta_\tau(p) \neq 0$	Pour chaque pixel
Si $V_{\tau-1}(p) < N \times \lvert\Delta_\tau(p)\rvert$, $V_\tau(p) = V_{\tau-1}(p) + 1$	
Si $V_{\tau-1}(p) > N \times \lvert\Delta_\tau(p)\rvert$, $V_\tau(p) = V_{\tau-1}(p) - 1$	

La dernière et la plus simple étape, consiste à comparer les deux critères de vraisemblance Δ_t et V_t. On considère que le pixel est en mouvement dans le cas où $\Delta_t > V_t$, et qu'il est fixe dans

[2] Activité temporelle : changement par rapport au temps (*mouvement*)

le cas inverse.

$\forall \tau \in [0, t_{final}]$	Pour chaque trame
$\forall p = (x, y) \in [(1, 1), (largeur, hauteur)]$	Pour chaque pixel
$V_0(p) = \Delta_0(p)$	Initialisation
Si $\Delta_\tau(p) > V_\tau(p)$, $D_\tau(p) = 1$	
Si $\Delta_\tau(p) < V_\tau(p)$, $D_\tau(p) = 0$	

3.5.3 Stratégie de régularisation spatio-temporelle (*Rebouclage*)

On présente une autre stratégie de régularisation, qui permet une meilleure détection, et d'éliminer ou de réduire les trois effets non voulus :

1. Les pixels *non significatifs* (bruit, fausse détection) ;
2. *L'effet fantôme*, qui produit une mauvaise détection lors du mouvement de l'objet après une longue durée en repos ;
3. *L'effet d'ouverture*, qui cause une détection médiocre des objets qui ont une faible projection du mouvement, comme par exemple *Les objets qui bougent radialement.*

Reconstruction hybride des contours

Dans cette première partie, on a des opérations de traitement à niveaux de gris, les deux entrées sont l'images I_t ainsi que la différence Δ_t. Le but de ce module est d'éliminer les objets fantômes de Δ_t, tout en les discriminant à l'intérieur de I_t, alors l'opération est :

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

Cette fonction permet de reconstruire les objets en mouvement à partir des contours, ces derniers sont obtenus en calculant la grandeur $Min(\|\nabla(I_t)\|, \|\nabla(\Delta_t)\|)$. Si on utilise une reconstruction géodésique classique $Rec^Y(X)$ basée sur la morphologie classique, alors on obtient dans

la plupart des cas, une reconstruction de l'objet avec son fantôme, c'est pour cela qu'on utilise la reconstruction hybride $HRec^J_\alpha(I)$ [RIC06][MAN05a], basée sur la morphologie oublieuse. L'avantage de cette dernière reconstruction est qu'elle oublie graduellement le marqueur

La morphologie binaire spatio-temporelle

Après le calcul de la reconstruction hybride, on utilise Δ'_t au lieu de Δ_t dans la suite. On compare alors Δ'_t avec V_t, on obtient ainsi la valeur de la détection D_t. On la régularise en éliminant ces petits composantes par l'ouverture par reconstruction :

$$L_t^{(0)} = Rec^{D_t}(\epsilon(D_t))$$

ϵ : est l'érosion morphologiques.

Enfin on fait une confirmation temporelle en calculant une deuxième reconstruction :

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

Mise à jour de la moyenne avec rebouclage

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 $\Sigma - \Delta$ seulement lorsque $L_t(x) = 0$.

Donc le nouveau algorithme obtenu est :

a) Image originale

b) La moyenne $\Sigma - \Delta$

c) La variance $\Sigma - \Delta$

d) La détection brute $\Sigma - \Delta$

FIG. 3.6 – L'estimateur $\Sigma - \Delta$ avec $N = 4$

•Différence signée
$S_t = M_t - I_t$
•Mise à jour de la différence
si $|S_t| \neq 0$:
  si $V_t < N \cdot |S_t|, V_t = V_{t+1} + 1$
  si $V_t > N \cdot |S_t|, V_t = V_{t-1} - 1$
•Reconstruction hybride des contours
$\Delta'_t = HRec_{\alpha}^{|S_t|}(Min(\|\nabla(I_t)\|, \|\nabla(|St|)\|))$
•Détection temporelle
si $\Delta'_t < V_t$
  alors $D_t = 0$
  sinon $D_t = 1$
•Traitement binaire spatio-temporel
$L_t^{(0)} = Rec^{D_t}(\epsilon(D_t))$
$L_t = Rec^{L_t^{(0)}}(L_{t-1}^{(0)})$
•Mise à jour de la moyenne avec rebouclage
si $L_t = 0$
  si $S_t < 0, M_t = M_{t-1} + 1$
  si $S_t > 0, M_t = M_{t-1} - 1$

D'après ce qu'on a vu précédemment, cette méthode présente une robustesse en terme de

détection, ainsi qu'une faible consommation de ressources[3]. Malgré ces avantages, nous avons des limitations dans la capacité d'adaptation à certaines scènes complexes, comme par exemple pour un mouvement périodique de forte amplitude, il sera mal interprété, d'où la nécessité de changer la période d'échantillonnage, afin d'obtenir une estimation plus riche quantitativement de l'activité du mouvement[RIC06].

3.6 Conclusion

Dans ce chapitre nous avons citer la morphologie temporelle qui consiste à rechercher le maximum et le minimum, on a besoin donc de stocker les images en question et les comparer pour avoir le gradient, la mémoire allouée dépend du choix de τ, cette méthode est considéré comme étant la plus excessive en terme d'occupation mémoire. A cause de l'insuffisance de cette méthode nous avons penser à la morphologie oublieuse, cette méthode apporte une segmentation, une visualisation de l'objet en mouvement nettement meilleurs et un bruit mieux filtré.

La moyenne récursive occupe moins de taille mémoire puisque il s'agit d'une utilisation d'un buffer de taille deux fois la taille de l'image, mais il y aura tout un calcul qui va se faire. Pour cela on a proposé de choisir un α qui est une puissance de 2 négative pour annuler les cycles de calcul de la multiplication qui risque de retarder le traitement sur le matériel. Cette solution nous permet de faire seulement un décalage à droite qui est nettement plus rapide qu'une multiplication classique.

L'estimateur $\Sigma - \Delta$ qui se base sur une convergence (incrémentation - décrémentation) est la plus précise que toutes les autres méthodes étudiées dans ce mémoire.

On classe les différentes méthodes vues précédemment par ordre décroissant en terme d'allocation de la mémoire :

– La gradient temporel puisque il stocke les images suivant le choix de τ. par exemple si

[3]ressources : mémoire et calculateur

$\tau = 4$, on stocke 4 images.

- L'estimateur $\Sigma - \Delta$ puisqu'on stocke en mémoire $I_\tau, M_\tau, \Delta_\tau, V_\tau$.
- Le gradient oublieux et la moyenne récursive occupent moins d'espace mémoire puisque ils stockent l'image courante et le fond précédent.

4

Les circuits FPGA

4.1 Introduction

Les composants logiques programmables sont traditionnellement regroupés au sein de deux familles : les CPLD et les FPGA.

- Les premiers sont des PLD (Programmable Logic Device) complexes, car ils sont composés de plusieurs macro cellules programmables simples, réparties autour d'une matrice d'interconnexion.
- Les FPGA (Field Programmable Gate Array) présentent une mer de modules logiques de petite taille, noyés dans un canevas de routage.

La confusion ayant pu régner par le passé entre des CPLD et des FPGA, c'est à présent le mode d'interconnexion qui tend à déterminer le type de PLD. Cette distinction est d'ailleurs la plus significative pour l'utilisateur. En effet, les temps de propagation étant plus prévisibles pour les CPLD du fait de leur schéma d'interconnexion, c'est cette propriété qui permettra de faire un choix quant au type du circuit logique à privilégier pour une application donnée.

4.2 Définition

Les FPGA (Field Programmable Gate Array ou *Réseaux Logiques Programmables*) sont des circuits reconfigurables ce qui leur donnent une très grande souplesse d'où la possibilité de les reprogrammer à volonté en un temps très court.

Le progrès de ces technologies permet de faire des composants toujours plus rapides et à plus haute intégration, ce qui permet de programmer des applications importantes. Cette technologie permet d'implanter un grand nombre d'applications et offre une solution d'implantation matérielle à faible coût pour des compagnies modestes, car le coût de développement d'un circuit intégré spécifique implique un trop lourd investissement.

Les utilisations sont nombreuses, on en cite :

- Prototypage de nouveaux circuits (*nouveaux microprocesseurs pour ordinateur ...*).
- Fabrication de composants spéciaux en petite série.
- Dans les systèmes embarqués de commande temps réel : l'avionique ; chaque carte de commande de vol d'un Airbus emploie plusieurs FPGA ACTEL. Les routeurs de réseau informatique emploient plusieurs FPGA ; les systèmes de l'informatique industrielle aussi.
- Adaptation aux besoins rencontrés lors de l'utilisation (*par exemple dans des satellites ou des sondes spatiales*).

D'une manière plus prospective, on se rapproche de la vie artificielle avec une intelligence capable d'apprentissage et d'adaptabilité. On trouve déjà des FPGA dans des éléments de transmission haut débit (ALCATEL, CISCO ...).

4.3 Architecture du circuit FPGA[VER06]

Un circuit FPGA est constitué des trois éléments principaux : une grande surface qui contient un grande quantité de blocs logiques entourée par plusieurs blocs entrée-sortie reliés tous par un réseau d'interconnexions programmable.

L'architecture d'un FPGA dépend principalement du fabricant (XILINX, ALTERA, ...), pour XILINX elle se présente sous la forme de deux couches :

- **U**ne couche appelée circuit configurable : constituée d'une matrice de blocs logiques configurables CLB permettant de réaliser des fonctions combinatoires et des fonctions séquentielles. Tout autour de ces blocs logiques configurables, nous trouvons des blocs entrées-sorties IOB dont le rôle est de gérer les entrées-sorties réalisant l'interface avec les modules extérieurs.
- **U**ne couche réseau mémoire SRAM : qui mémorise tous les configuration, les interconnexion et les fonctions réalisées par les blocs logiques.

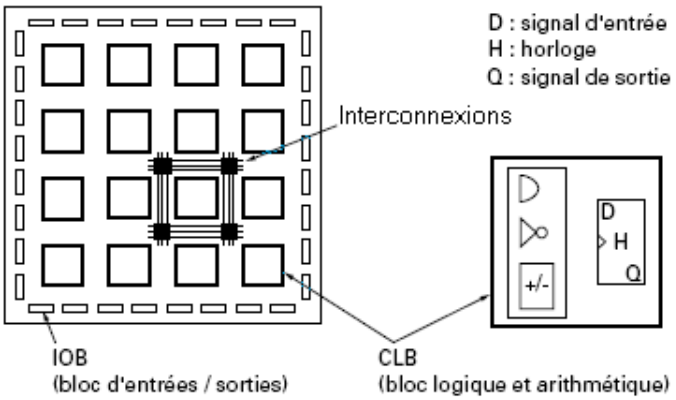


FIG. 4.1 – Architecture d'un circuit FPGA

4.4 Les CLB (*configurable logic block*)[VER06]

On peut déterminer les performances d'une FPGA à partir des CLB. Chaque CLB est composé de deux générateurs de fonctions logiques combinatoires à 4 entrées et une seule sortie et de deux bascules D qui permettent la mémorisation ainsi que la synchronisation. On a aussi 4 autres entrées qui font la connexion interne entre les différents éléments du bloc (FIG-4.2).

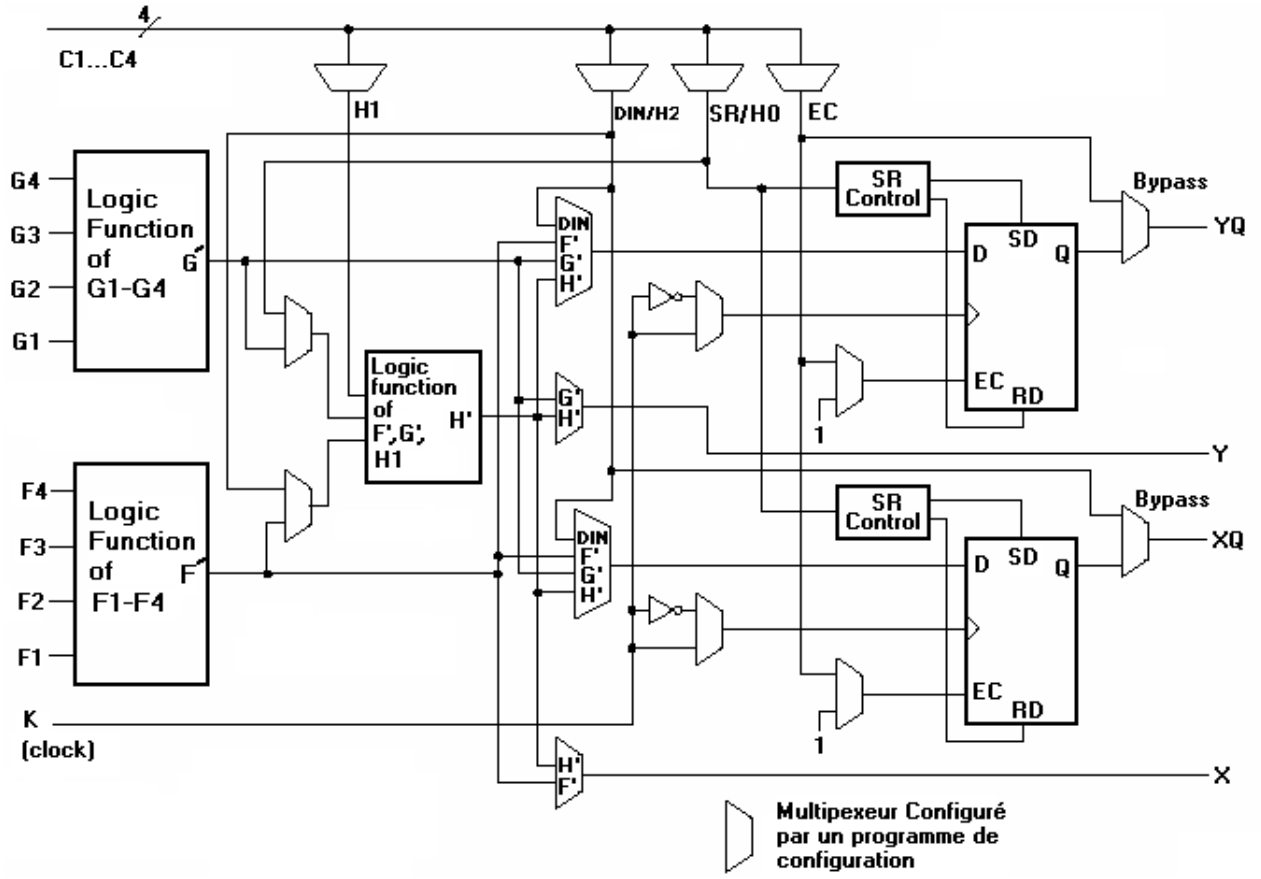


FIG. 4.2 – Le schéma d'un bloc logique configurable CLB

On commence par les générateurs de fonctions F' et G', chacune a quatre entrées indépendantes (F1,F2,F3,F4) et (G1,G2,G3,G4) d'où une flexibilité de développement, la table de vérité des deux fonctions est inscrite dans une zone de mémoire, d'où on a toujours une lecture de la table de vérité ce qui signifie que le temps de propagation ne dépend pas de la fonction générée. On a aussi une troisième fonction générée à partir des deux fonctions précédentes ainsi que l'entrée H1 qui appartient aux signaux de contrôle. On peut avoir des sorties directes non synchronisées, une sortie X à partir de F' et H' et une sortie Y à partir de G' et H'. D'après toutes ces possibilités on peut générer :

– Deux fonctions indépendantes à quatre variables indépendantes.

– Une fonction à cinq variables.

– Deux fonctions une à 4 variables et l'autre à cinq.

Les sorties des générateurs de fonctions peuvent être aussi appliquées à des bascules. Chaque bascule présente deux modes de fonctionnement :

1. Un mode *flip-flop* qui mémorise, soit l'une des fonctions F', G', H', soit l'entrée directe DIN. La donnée peut être mémorisée sur un front montant ou descendant de l'horloge (CLK). Les sorties de ces deux bascules correspondent aux sorties du CLB XQ et YQ.
2. Un mode dit de *verrouillage* exploite une entrée S/R qui peut être programmée soit en mode SET, mise à 1 de la bascule, soit en Reset, mise à zéro de la bascule. Ces deux entrées coexistent avec une autre entrée appelée le global Set/Reset. Cette entrée initialise le circuit FPGA à chaque mise sous tension, à chaque configuration, en commandant toutes les bascules au même instant soit à '1', soit à '0'. Elle agit également lors d'un niveau actif sur le fil RESET lequel peut être connecté à n'importe quelle entrée du circuit FPGA.
3. Un mode optionnel des CLB est *la configuration en mémoire* RAM de 16x2bits ou 32x1bit. Les entrées F1 à F4 et G1 à G4 deviennent des lignes d'adresses sélectionnant une cellule mémoire particulière. La fonctionnalité des signaux de contrôle est modifiée dans cette configuration, les lignes H1, DIN et S/R deviennent respectivement les deux données D0, D1 (RAM 16x2bits) d'entrée et le signal de validation d'écriture WE. Le contenu de la cellule mémoire (D0 et D1) est accessible aux sorties des générateurs de fonctions F' et G'. Ces données peuvent sortir du CLB à travers ses sorties X et Y ou alors en passant par les deux bascules.

4.5 Les IOB (*input output block*)[VER06]

Les blocs entrée/sortie sont des blocs intermédiaires entre les broches du composant FPGA et la logique interne développée à l'intérieur du composant. Ils sont présents sur toute la périphérie du circuit FPGA. Chaque bloc IOB contrôle une broche du composant et il peut

être défini en entrée, en sortie, en entrée/sortie ou inutilisé (haute impédance).

1. La configuration en entrée

   Le signal d'entrée traverse un buffer qui selon sa programmation peut détecter soit des seuils TTL ou soit des seuils CMOS. Il peut être routé directement sur une entrée directe de la logique du circuit FPGA ou sur une entrée synchronisée. Cette synchronisation est réalisée à l'aide d'une bascule de type D. Le changement d'état peut se faire sur un front montant ou descendant. De plus, cette entrée peut être retardée de quelques nanosecondes pour compenser le retard pris par le signal d'horloge lors de son passage par l'amplificateur. Le choix de la configuration de l'entrée s'effectue grâce à un multiplexeur (program controlled multiplexer). Un bit positionné dans une case mémoire commande ce dernier.

2. La configuration en Sortie

   Nous distinguons les possibilités suivantes :

   – Inversion ou non du signal avant son application à l'IOB.

   – Synchronisation du signal sur des fronts montants ou descendants d'horloge.

   – Mise en place d'un " pull-up " ou " pull-down " dans le but de limiter la consommation des entrées/sorties inutilisées,

   – Signaux en logique trois états[1] ou deux états. Le contrôle de mise en haute impédance et la réalisation des lignes bidirectionnelles sont commandés par le signal de commande `Out Enable` lequel peut être inversé ou non. Chaque sortie peut délivrer un courant de 12 mA. Ainsi toutes ces possibilités permettent au concepteur de connecter au mieux une architecture avec les périphériques extérieurs.

[1] état haut (1), état bas (0), haute impédance (Z)

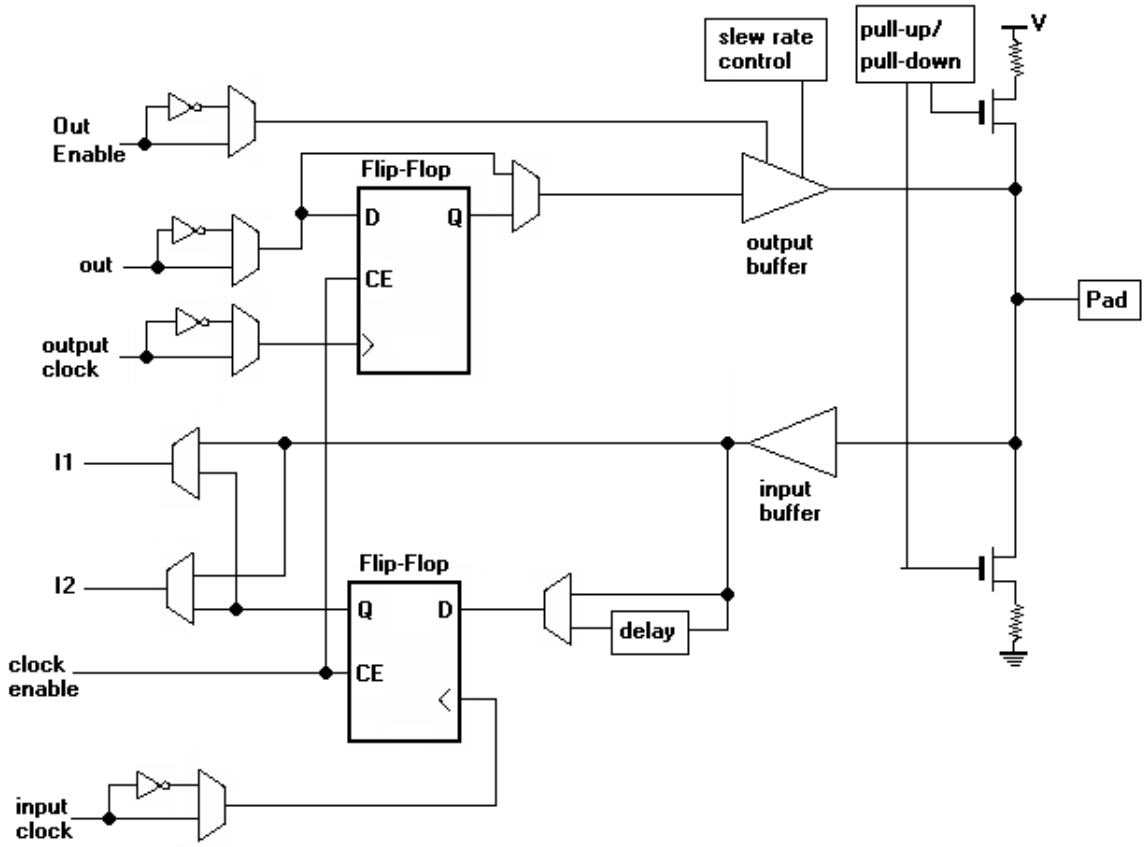


FIG. 4.3 – Le schéma d'un bloc Entrée/Sortie IOB

4.6 Les Interconnexion

Les connexions internes dans les circuits FPGA sont composées de segments métallisés. Parallèlement à ces lignes, nous trouvons des matrices programmables réparties sur la totalité du circuit, horizontalement et verticalement entre les divers CLB. Elles permettent les connexions entre les diverses lignes, celles-ci sont assurées par des transistors MOS dont l'état est contrôlé par des cellules de mémoire vive ou RAM. Le rôle de ces interconnexions est de relier les blocs logiques et les entrées/sorties avec un maximum d'efficacité, afin que le temps d'exécution dans le circuit soit le moins élevé possible. Pour parvenir à cet objectif, XILINX propose trois sortes d'interconnexions selon la longueur et la destination des liaisons.

1. Interconnexions à usage général.

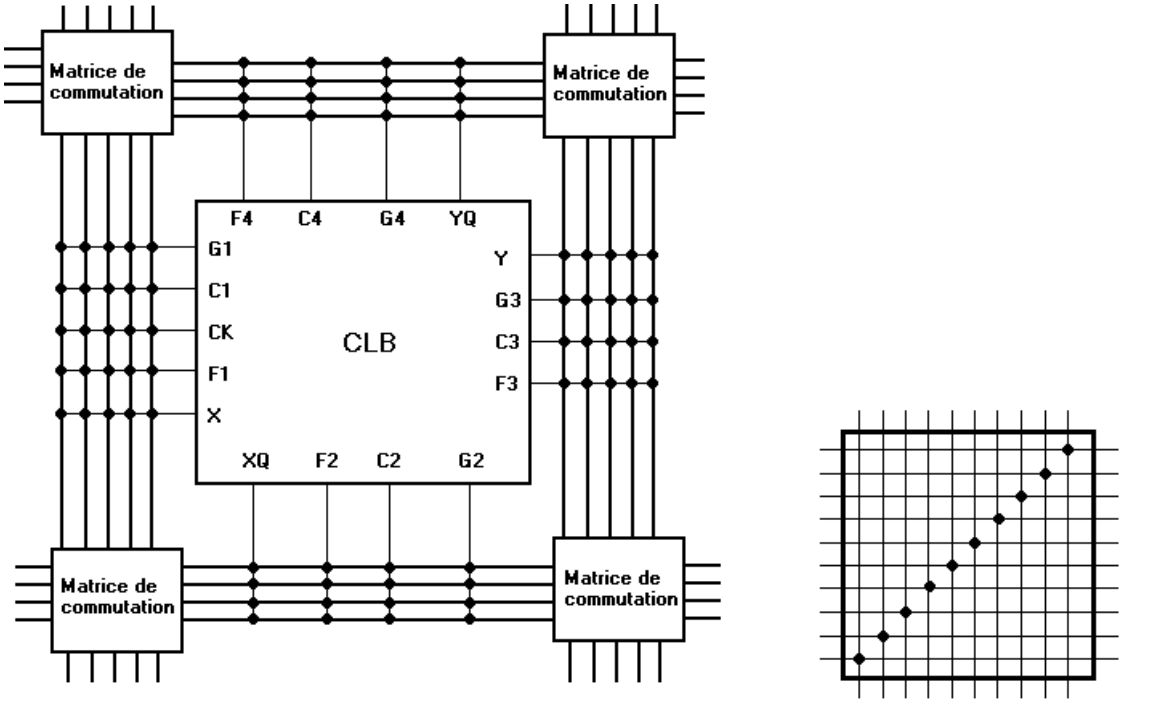
2. Interconnexions directes.

3. Les longues lignes.

4.6.1 Interconnexions à usage général

Ce système fonctionne en une grille de cinq segments métalliques verticaux et quatre segments horizontaux positionnés entre les rangées et les colonnes de CLB et de l'IOB.

Des aiguilleurs appelés aussi matrices de commutation sont situés à chaque intersection. Leur rôle est de raccorder les segments entre eux selon diverses configurations, ils assurent ainsi la communication des signaux d'une voie sur l'autre. Ces interconnexions sont utilisées pour relier un CLB à n'importe quel autre. Pour éviter que les signaux traversant les grandes lignes ne soient affaiblis, nous trouvons généralement des buffers implantés en haut et à droite de chaque matrice de commutation.



a)Les interconnexions b)La matrice de commutation

FIG. 4.4 – Les Connexions à usage général

4.6.2 Les interconnexions directes

Ces interconnexions permettent l'établissement de liaisons entre les CLB et les IOB avec un maximum d'efficacité en terme de vitesse et d'occupation du circuit. De plus, il est possible de connecter directement certaines entrées d'un CLB aux sorties d'un autre.

Pour chaque bloc logique configurable, la sortie X peut être connectée directement aux entrées C ou D du CLB situé au-dessus et les entrées A ou B du CLB situé au-dessous. Quant à la sortie Y, elle peut être connectée à l'entrée B du CLB placé immédiatement à sa droite. Pour chaque bloc logique adjacent à un bloc entrée/sortie, les connexions sont possibles avec les entrées I ou les sorties O suivant leur position sur le circuit.

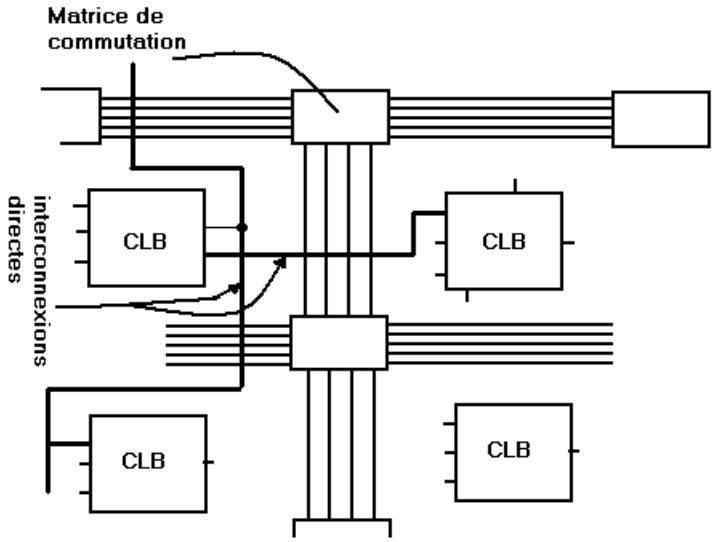


FIG. 4.5 – Les interconnexions directes

4.6.3 Les longues lignes

Les longues lignes sont de longs segments métallisés parcourant toute la longueur et la largeur du composant, elles permettent éventuellement de transmettre avec un minimum de retard, les signaux entre les différents éléments dans le but d'assurer un synchronisme aussi parfait que possible. De plus, ces longues lignes permettent d'éviter la multiplicité des points

d'interconnexion.

4.7 Conclusion

Ce chapitre nous a permis de donner une descriptions des caractéristiques essentielles des FPGA, en abordant leur architecture et leurs configurations. A partir de cette étude, nous avons jugé que les circuits FPGA sont très performants vu les nombreux avantages qu'ils présentent, tels que : la souplesse de programmation et la possibilité d'une reconfiguration dynamique du circuit.

Deuxième partie

Conception et réalisation

5

Implémentation sur FPGA

5.1 Introduction

Dans ce chapitre nous allons présenter les différentes versions de la carte RC203, En particulier nous étudierons la carte RC203E. Commençons par la plateforme de développement PDK sur ces deux aspects (Software et Hardware), en passant par les différentes étapes de la procédure de programmation et enfin l'implémentation de la routine.

5.2 Etude de la carte RC203E [BS03]

La carte RC203E est une plateforme pour l'évaluation et le développement de haute performance basée sur FPGA . Cette plateforme contient un circuit FPGA Xilinx Virtex-II, une mémoire externe, des horloges programmables, Ethernet, Audio, Vidéo, SmartMedia, Port parallèle, RS-232 et PS/2 pour le clavier et la souris. La carte RC203 existe sous trois versions :

- Standard (RC-I-203-2V3K4S) ;
- Professionnelle (RC-I-203-2V3K4P) ;
- Expert (RC-I-203-2V3K4E).

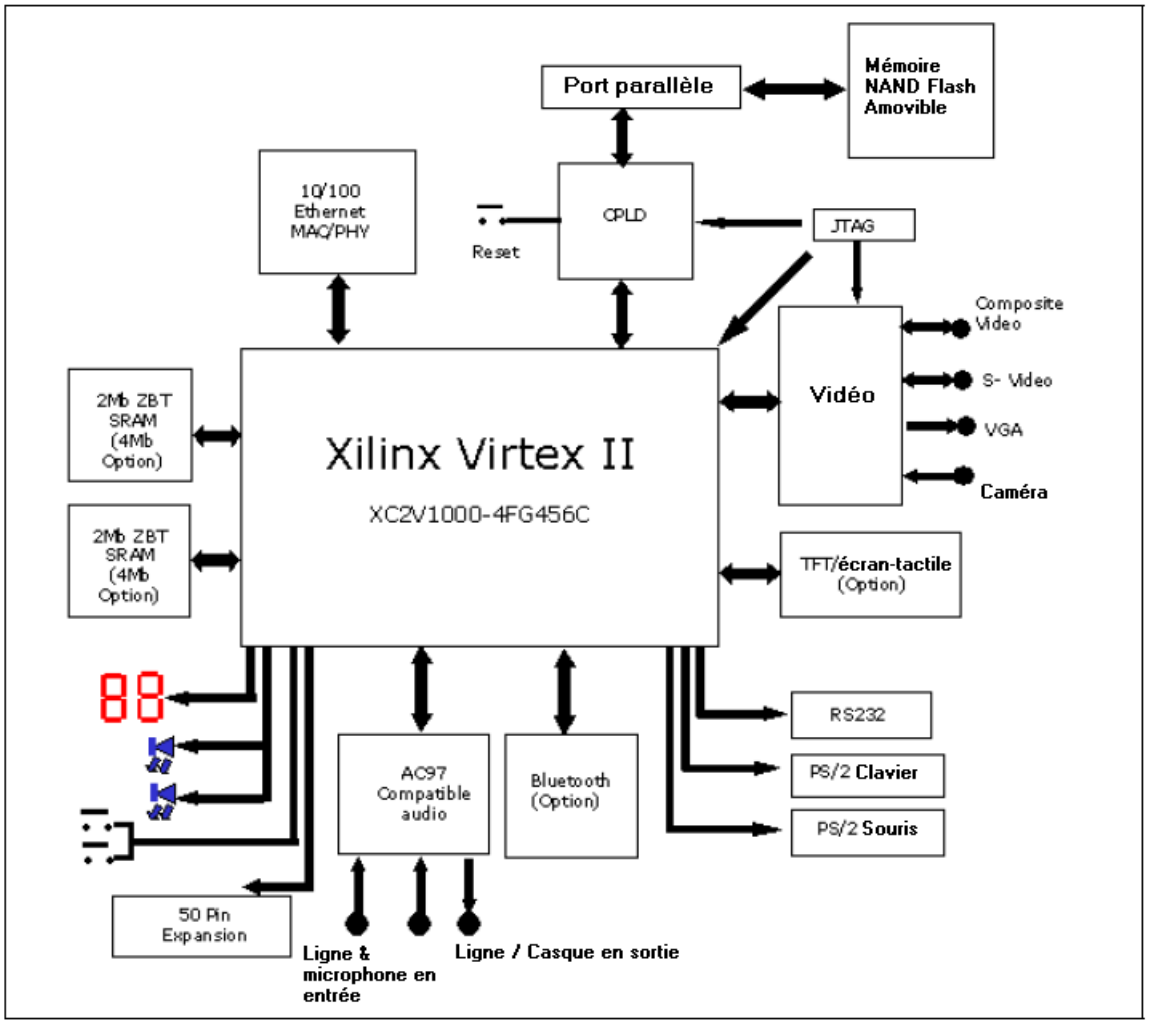


FIG. 5.1 – Le schéma bloc de la carte RC203

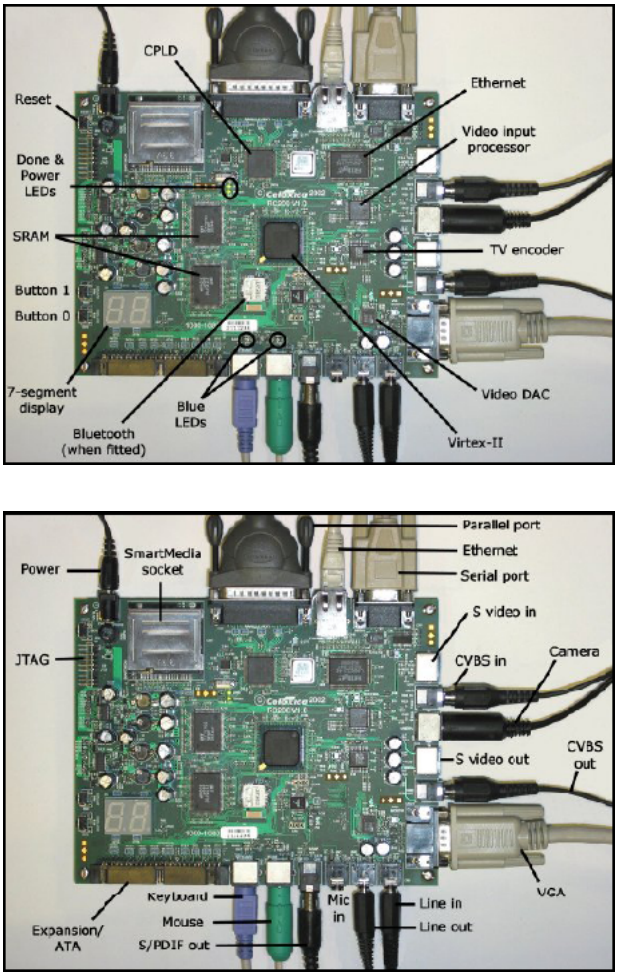


FIG. 5.2 – La carte RC203E

5.2.1 Les différents accessoires de la carte RC203

Comme le montre les deux figures FIG 5.1 et FIG 5.2 la carte RC203E est très riche en accessoires, mais cette richesse diffère d'une version à l'autre.

La version *Standard* RC203S

C'est la version de base elle comporte :

1. FPGA VIRTEX-II 2V3000-4
2. Ethernet MAC/PHY with 10/100baseT socket

3. 2 ZBT[(1)] SRAM chacune est de taille de 2-MB
4. Support vidéo qui contient
   (a) Composite video E/S
   (b) S-Video E/S
   (c) VGA en sortie
   (d) Camera en entrée
5. Support audio compatible avec AC'97 qui contient
   (a) Microphone en entrée
   (b) Ligne (Stéréo) en entrée
   (c) Ligne/Casque (Stéréo) en sortie
6. Connecteur mémoire flash SmartMedia pour le stockage des fichier BIT
7. CPLD pour la configuration/reconfiguration et la gestion SmartMedia
8. Connecteur parallèle, pour télécharger et héberger les fichier BIT dans l'FPGA
9. Port série RS-232
10. PS/2 connecteurs clavier et souris
11. 2 afficheurs sept-segments
12. 2 LEDs bleus
13. 2 switches de contact instantanés
14. 50 pin (33 E/S, 3 alimentations, 2 horloges)
15. Connecteurs JTAG

[1]Zero Bus Turnaround : Changement d'accès de la lecture à l'écriture ou l'inverse en zéro cycles d'horloges

16. Couvertes perspex en haut et en bas

17. Alimentation universelle 110/240V

La version *Professionnelle* RC203P

C'est une version un peu plus évoluée que sa précédente, elle comporte en addition de la version standard :

1. Un micro-casque

2. Souris

3. carte SmartMedia 16MB

4. Caméra couleur

La version *Expert* RC203E

C'est la plus évoluée, elle a en addition de la version professionnelle :

1. Module sans-fil Bluetooth

2. 2 ZBT SRAM de taille de 4-MB (et non 2-MB)

3. Affichage TFT ou écran-tactile

5.2.2 La plateforme de développement PDK

Celoxica a mis en point un outil, qui permet le développement d'algorithmes ainsi que le contrôle de leur exécution et la gestion du software et hardware.

1. **Software (DK Design Suite)** :

   DK Designe suite, permettant de programmer la carte en utilisant le langage de programmation *handle-C*. Ce dernier est similaire au C++ en terme de syntaxe, sauf qu'il

comporte quelques nouvelles commandes, comme par exemple `par` (pour indiquer que les instruction doivent s'exécuter en parallèle), `seq` (qui veut dire que les instruction vont s'exécuter séquentiellement), ···

2. **Hardware** :

   FTU2[2] qui permet de charger le fichier `BIT` (*le fichier binaire contenant le programme*) dans le circuit FPGA , en d'autres termes implémenter le circuit FPGA .

5.2.3 Procédure de programmation de la carte RC203E

La partie *software*

Avant de commencer le développement il faut toujours créer un projet, d'où on doit passer par les deux étapes suivantes :

- **Etape 1** : dans le menu `File > New (Ctrl+N)`, La boite de dialogue suivante FIG : 5.3 va apparaître, on sélectionne `Chip`, et on donne ainsi un nom au projet (`PRJ` *dans notre cas*), on appuie en suite sur `OK`.
- **Etape 2** : de la même façon on crée un fichier source, Dans le menu `File > New (Ctrl+N)`, mais cette fois-ci on doit choisir `Source file`, on sélectionne `Handle-C Source File` et on lui donne un nom (`PROG` *dans notre cas*), on appuie aussi sur `OK`.

[2] File Transfer Utility

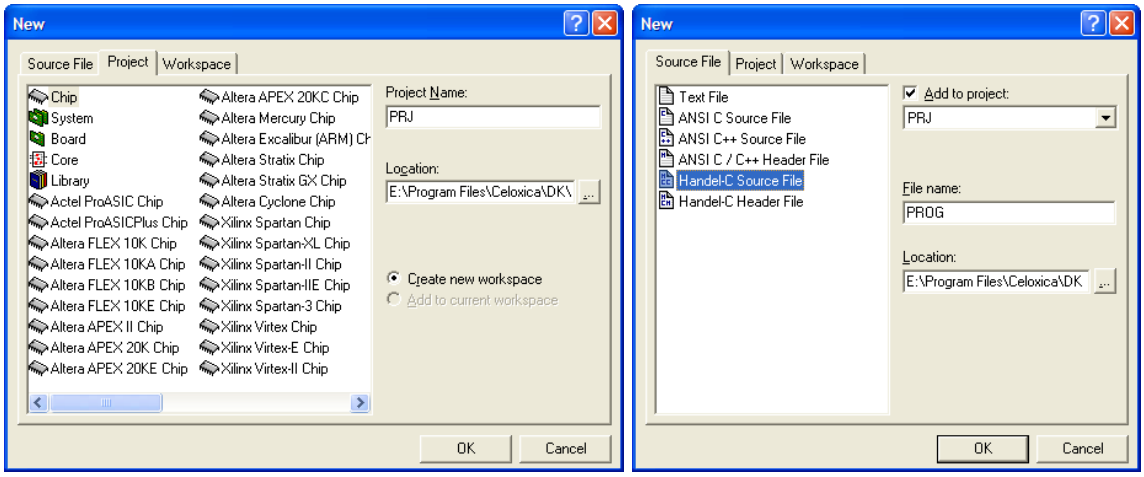


FIG. 5.3 – La boite de dialogue d'un nouveau Projet/Fichier Source

On obtient par la suite un projet vide prêt à utilisé.

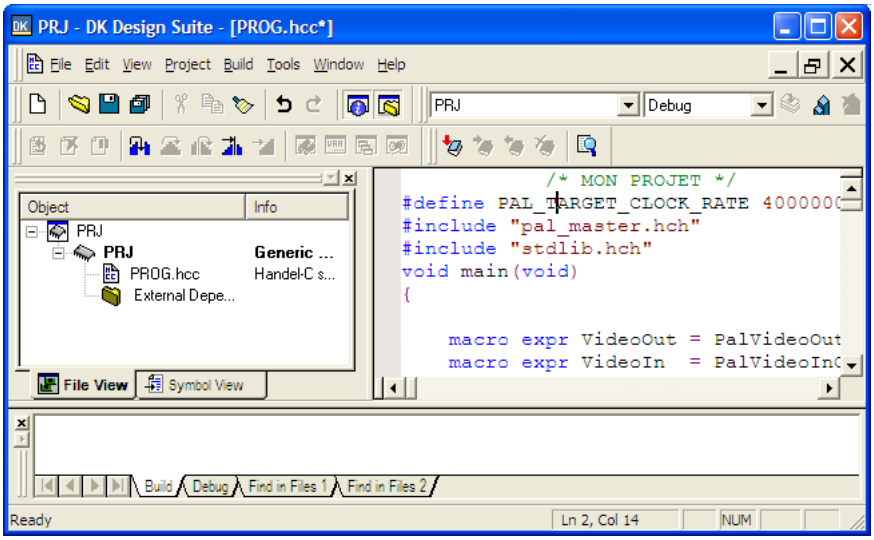


FIG. 5.4 – L'environnement de développement

Ensuite, nous devons configurer notre projet, dans notre cas on a deux types de configurations, la *simulation* et la création du fichier `EDIF` qui nous aidera par la suite à créer le fichier `bit`, on peut ajouter une nouvelle configuration dans le menu `Build > Configurations`, et de changer ces paramètres dans le menu `Project > Settings (Alt+F7)`

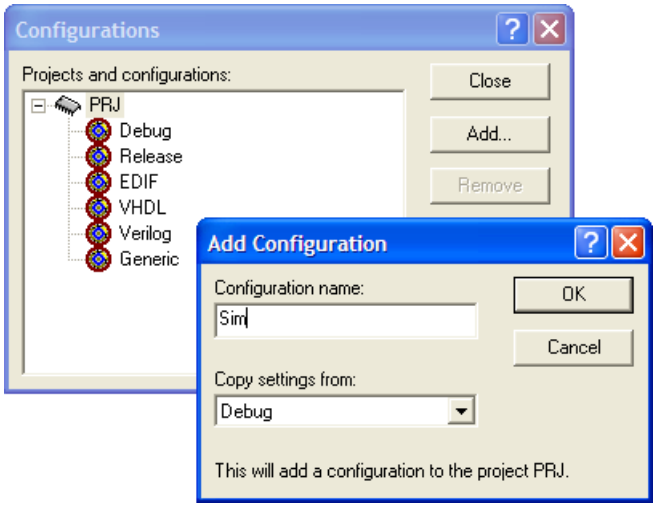


FIG. 5.5 – Création d'une nouvelle configuration

1. **La simulation** : pour la simulation les paramètres doivent être copiés à partir de ceux de `Debug`, ensuite on change dans "la boite de dialogue des paramètres" les éléments suivants :
   - dans `General` on doit renommer les fichiers intermédiaires ainsi que le répertoire de sortie (`Sim` *dans notre cas*).
   - dans `Preprocessor` la définition des préprocesseur doit être : `NDEBUG,USE_SIM,__EDIF__`.
   - dans `Optimizations` on doit cocher `Hight-level optimization (-O+hight)`.
   - dans `Chip` on met dans `part` le nom de la configuration (`Sim` *dans notre cas*).
   - dans `Linker` c'est tous comme le C++, là où on définit les bibliothèques. La première des choses est de définir les deux bibliothèques indiquant la simulation `sim.hcl`[3] et `pal_sim.hcl` dans `"Object/library modules"`, ensuite on ajoute toute bibliothèque utilisée dans le programme comme fichier entête à l'exception de `"pal_master.hch"`. Par exemple pour la souris on doit introduire l'entête `"pal_mouse.hch"` d'où on ajoute la bibliothèque `"pal_mouse.hcl"`. Dans `"Additional C/C++ Modules"` on introduit le chemin du fichier `PalSim.lib` qui est généralement `"Répertoire d'installation/`

[3] hcl : Handle-C library

`Celoxica/ PDK/ software/ lib/"`, et on coche enfin `"Save browser info"`.

2. **Le fichier EDIF de RC203E** : les paramètres cette fois-ci seront copiés à partir de ceux de `EDIF`, et on changera par la suite les éléments suivants :
   - dans `General` on renomme de la même manière que pour la simulation (`RC203E` *dans notre cas*).
   - dans `Preprocessor` on met `NDEBUG,USE_RC203E,__EDIF__`
   - dans `Synthesis` on choisit `"Speed(-N+speed)"`, `"Enable mapping to ALUs"`, `"MULT18X18 to 96"` et on coche tous les autres sauf `"disable fast carry chain optimizations"`
   - dans `Optimizations` on coche tous les cases sauf la dernière
   - dans `Chip` on choisit notre FPGA qui est XILINX VIRTEX-II (XilinxVirtexII), et bien sure on a `xc2v3000`, `fg676` et un ordre de vitesse de `"4"`.
   - dans `Linker` ici les deux bibliothèques obligatoires sont `"rc203e.hcl"` et `"pal_rc203e.hcl"` on ajoute comme pour la simulation toute bibliothèque utilisée. et on coche aussi `"save browser info"`

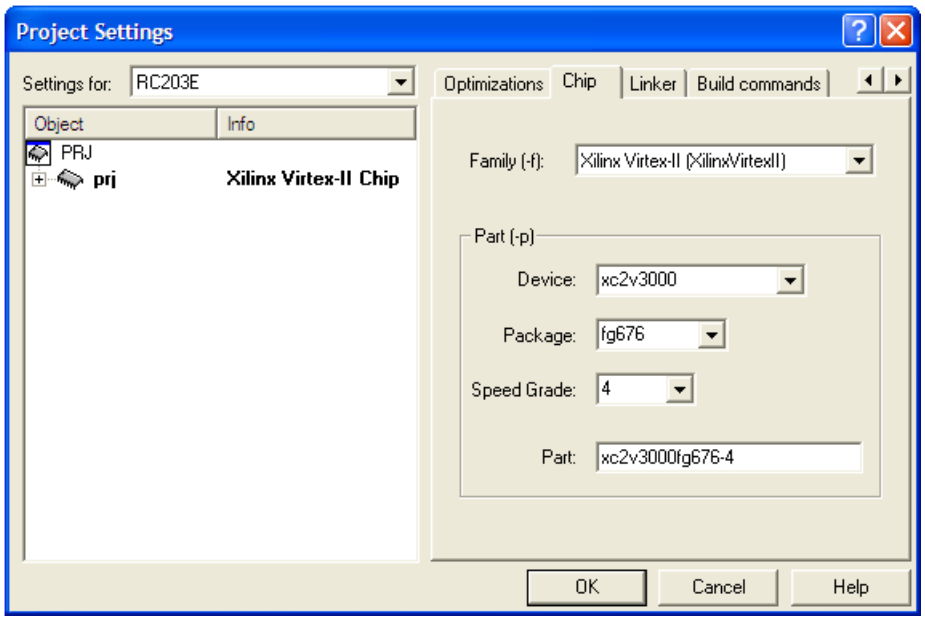


FIG. 5.6 – La boite de dialogue des paramètres

Après avoir configurer la simulation ainsi que la création du fichier `EDIF`, l'étape suivante est de simuler ainsi que générer le fichier `bit` contenant le programme.

La simulation : La plateforme de développement DK Design Suite offre un environnement de simulation qui nous aide à visualiser efficacement le fonctionnement du programme. Pour simuler il faut d'abord construire le projet d'où dans le menu `Build > Build prj (F7)` (`prj` *dans notre cas*) on peut construire notre projet, après sa construction on peut l'exécuter en allant dans le menu `Build > Start Debug (F5)`. Après l'exécution l'interface de simulation apparaîtra ce qui est montré dans FIG-5.7

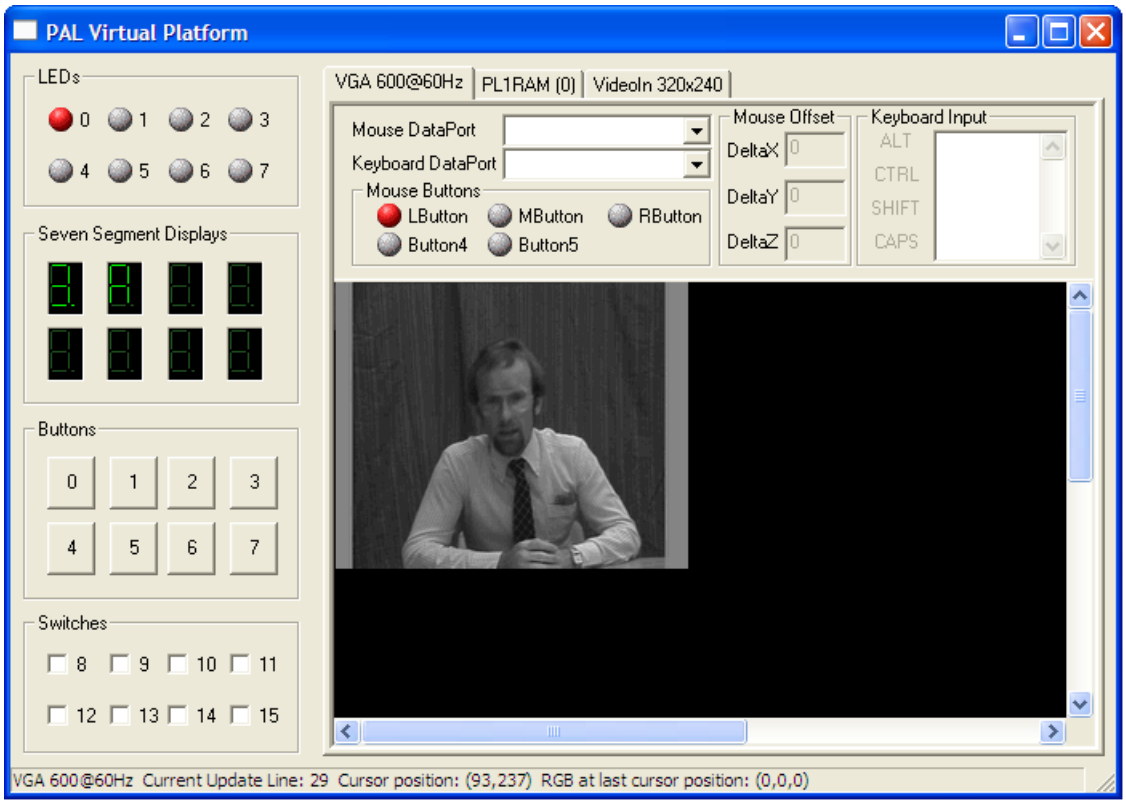


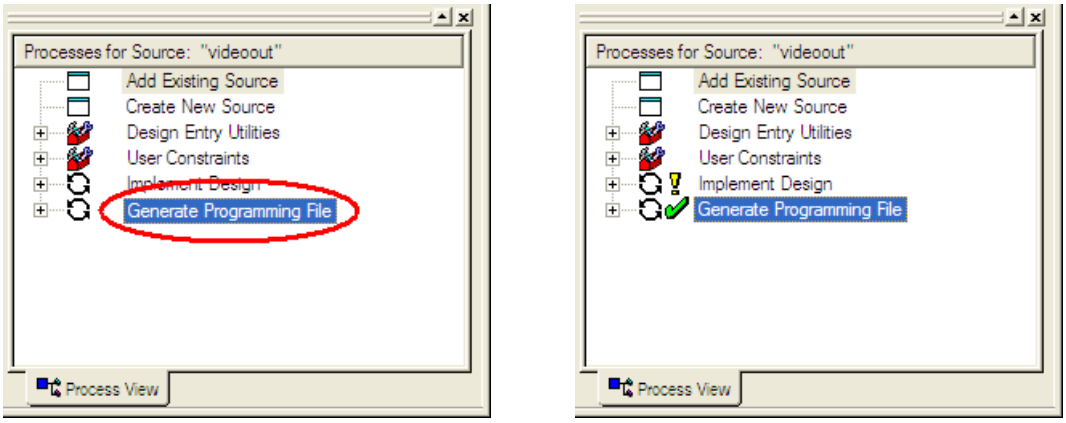
FIG. 5.7 – L'interface de simulation

La création du fichier `EDIF` : On peut remarquer clairement que si on choisit la configuration `RC203E`, le menu `Build > Start Debug (F5)` sera désactivé automatiquement, d'où dans cette étape il suffit juste de construire le projet afin d'avoir le fichier `EDIF` en sortie.

La création du fichier `bit` : Le fichier `EDIF` n'est pas un fichier binaire, il n'est pas aussi celui qui doit être implémenté sur le FPGA , sa génération n'est qu'une étape intermédiaire. Pour générer le fichier `bit` il faut faire appel à Xilinx Project Navigator, qui nous permet de transformer le fichier `EDIF` en un fichier `bit`, d'où on doit suivre les étapes suivantes :

- **Etape 1** : dans le menu `File > New Project` on crée un nouveau projet, on lui donne un nom ainsi qu'un chemin, on doit choisir aussi l'option "EDIF" dans le `combobox "Top-level Module Type"`, si on appuie sur `suivant`, une deuxième boite de dialogue apparaîtra, c'est là où on doit introduire le fichier `EDIF` à convertir, on appuie aussi sur `suivant`, une dernière chose reste à configurer, dans la boite de dialogue suivante on doit choisir le même type du FPGA que celui de la configuration `RC203E`, on clique sur `suivant` ensuite sur `terminer`.
- **Etape 2** : après que le projet soit généré, on fait un double clique sur `"Generate programming file"` comme le montre (Fig-5.8), quand la génération du fichier termine, le fichier `bit` sera prêt, et on le trouvera dans le répertoire du projet.

Après cette étape nous atteindrons la fin de la partie *software*



a) Début de la conversion b) Fin de la conversion

FIG. 5.8 – La conversion du ficher `EDIF` vers un fichier `bit`

La partie *hardware*

La chose qui nous reste après la partie précédente est de transférer le fichier `bit` au circuit FPGA , ce qui rend cette partie beaucoup plus simple que sa précédente. L'application qu'on va utiliser est Celoxica FTU2, elle nous permet d'implémenter le circuit FPGA en quelque petites étapes.

- **Etape 1** : on choisit le type de la carte ou la plus proche (RC200 *dans notre cas*),
- **Etape 2** : dans `"Port Settings"` on choisit le type de connexion port parallèle (0x378), port série (0x278) ou une autre adresse mémoire, et on teste la connexion en cliquant sur le bouton `"Test Settings"`
- **Etape 3** : dans `"FPGA"` on ouvre le fichier `bit` qu'on a déjà créé, on peut effacer la mémoire FPGA en cliquant sur `"Clear FPGA "` ou l'implémenter en cliquant sur `"Configurer"`

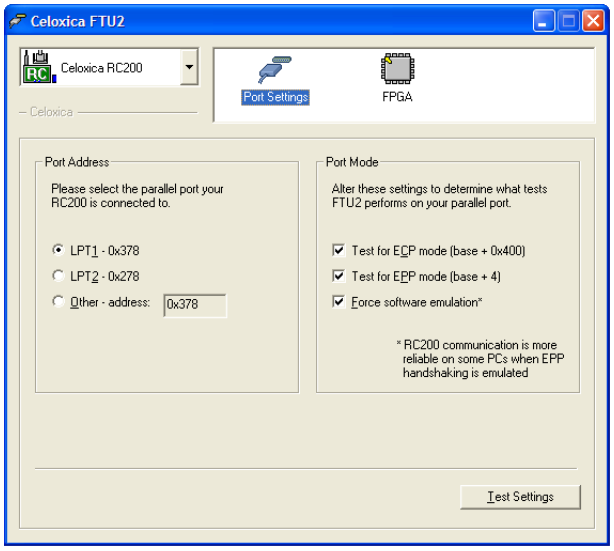


FIG. 5.9 – Configuration des ports

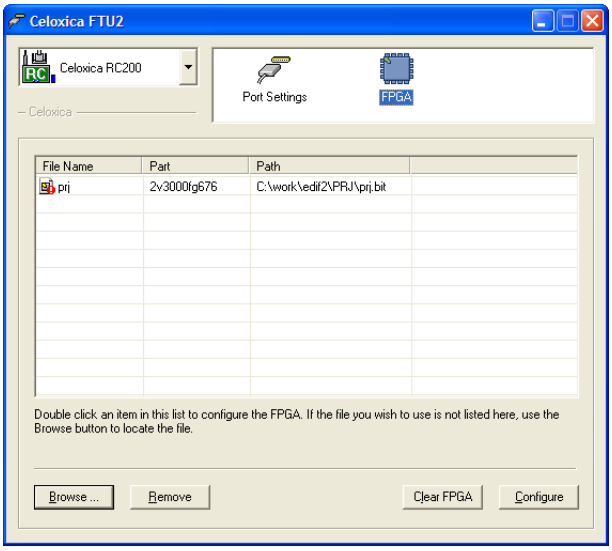


FIG. 5.10 – Implémentation sur FPGA

Après ces trois étapes le circuit FPGA commence à fonctionner automatiquement selon le programme implémenté.

5.3 Conception

5.3.1 Les différents blocs du système

La routine à implémenter sur le circuit FPGA consiste aux éléments suivants :

Vin2Ram Qui fait la lecture à partir d'une des entrées de la carte (S-video, caméra, ...). L'entrée `idx` spécifie l'indice de l'image à lire, cette dernière est stockée dans la mémoire `RAM1` à l'adresse `largeur`×`hauteur`×`idx`, affin d'éviter la redondance de stockage.

Gradient_temp, gradient_oub et moyrecc Ces trois bloc sont ceux qui calculent respectivement *le gradient temporel, le gradient oublieux* et *la moyenne récursive*. Ils ont comme entrées `RAM1` qui contient la séquence d'images à traiter, `idx` l'indice de l'image courante et `idy` qui représente le nombre d'images de la séquence. On a une sortie pour chaque bloc vers la mémoire `RAM2` avec

Filtre	`gradient_temp`	`moyrecc`	`gradient_oub`
Adresse dans `RAM2`	0	`h`×`w`×`idx`	2×`h`×`w`×`idx`

tel que : `h` et `w` sont respectivement la hauteur et la largeur de l'image.

Ram2Vout Qui affiche l'image traitée à la sortie (S-video, VGA, ...), ce bloc a comme entrées `RAM2, RAM1` qui contiennent respectivement l'image filtrée et l'image courante, `idx` pour l'affichage de l'image courante à partir de `RAM1`, `idz` avec lequel on choisit le filtre à utiliser et enfin `MousePtr` qui est une entrée de la souris pour l'interruption et l'arrêt de l'affichage, afin de pouvoir accéder les autres bloc comme `Vin2RAM, gradient_temp, ...`

5.3.2 Les entrées/sorties globales du système

VideoIn est une entrée à partir d'un téléviseur (*s-video, vidéo composite*), ou bien de la *caméra* équipée avec le la carte RC203E ,

Mouse est l'entrée PS/2 de la souris, qui permet de jouer les trois signaux intermédiaires `idx`, `idy` et `idz`.

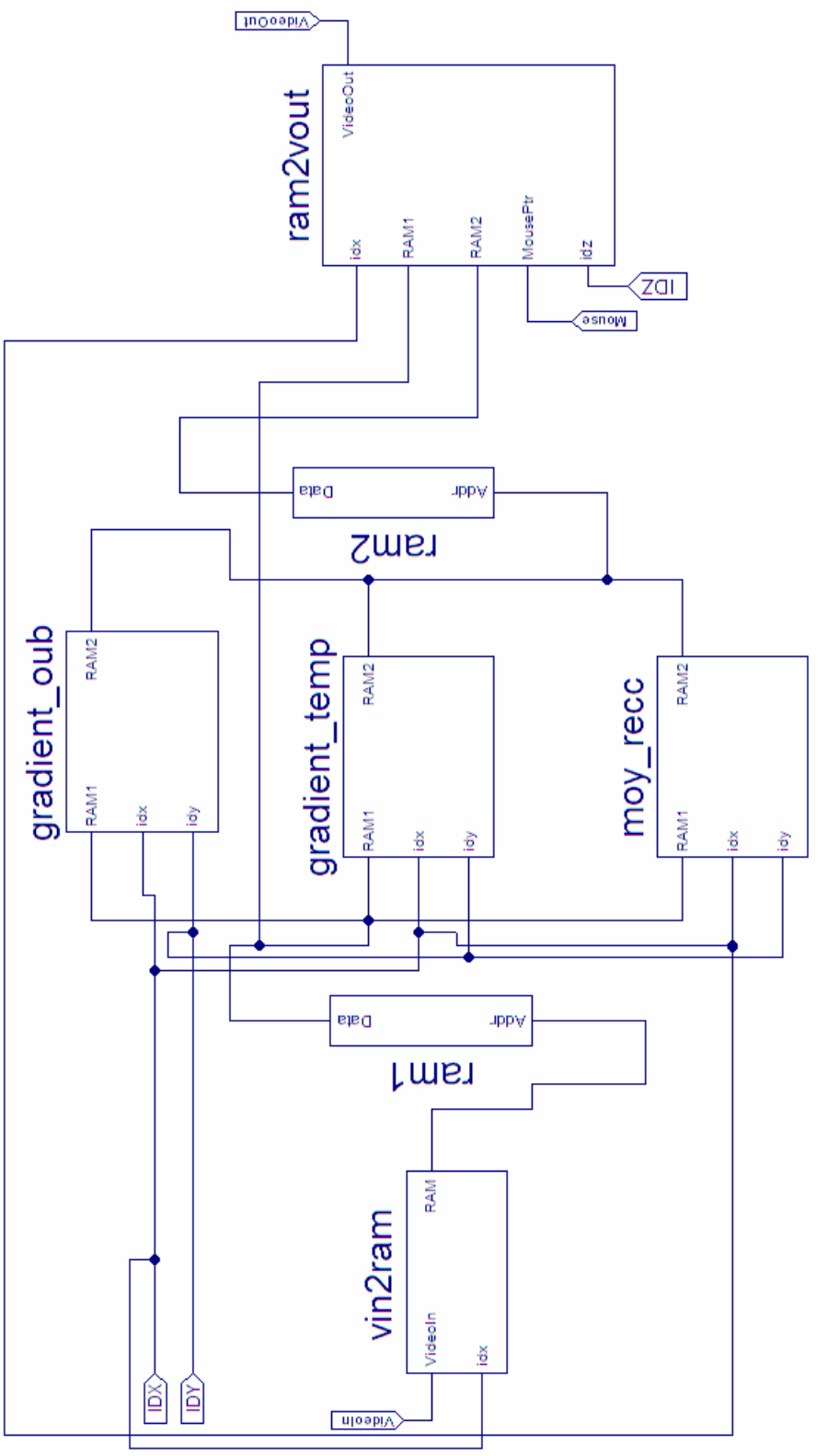


FIG. 5.11 – Le schéma bloc du circuit à implémenter sur FPGA

VideoOut permet d'afficher le résultat du filtrage sur un écran (*VGA*) ou bien sur un téléviseur (*s-video, vidéo composite*).

5.3.3 Les signaux intermédiaires

idx est un signal indiquant l'image courante à traiter ou à afficher sur la sortie vidéo (*VideoOut*), il peut être incrémenter en cliquant sur le bouton droit de la souris, ce signal prend des valeurs entre 0 et 7. Ces derniers sont affichées sur un afficheur 7-segments.

idy ceci indique le nombre total d'images lues à partir de *VideoIn*, si on clique sur le bouton gauche de la souris, ce signal sera incrémenté, de même `idy` prend des valeurs entre 0 et 7. Ces derniers sont affichées sur le deuxième afficheur 7-segments.

idz ce signal donne l'indice du filtre à utiliser (*gradient temporel, gradient oublieux* ou *moyenne récursive*), celui-ci est commandé par le bouton du milieu de la souris, et sa valeur est affiché sur les deux LED bleues, elle est ainsi comprise entre 0 et 3 (0 pas d'affichage).

Filtre	`gradient_temp`	`moyrecc`	`gradient_oub`
`idz`	1	2	3

5.3.4 Les ressources utilisées

Ressource	Utilisées	Disponibles	Taux d'utilisation
Bascules	2 223	14 336	15%
Bascules (FLIP-FLOP)	1 689	28 672	5%
LUT à 4 entrées	2 808	28 672	9%
IOB	199	484	41%
Horloges	3	16	18%

TAB. 5.1 – Le taux de ressources utilisées pour l'implémentation des trois algorithmes

D'après TAB - 5.1, on remarque bien que le cricuit FPGA XILINX VIRTEX II est largement suffisant pour l'implémentation de nos filtres. Comme on utilise directement la carte RC203E ainsi que le langage de description HANDLE-C, donc on ne peut pas déterminer exactement les ressources utilisées, car on passe par des bibliothèques prédéfinies, qui peuvent utiliser plus de

ressources, ainsi qu'à l'utilisation d'une caméra, une sortie VGA, souris, ... d'autres ressources vont être occupées automatiquement.

5.4 Simulation

Dans cette partie nous allons simuler le programme conçu précédement. Après le passage par la procédure 5.2.3, l'interface de simulation apparaitera Fig - 5.12. La procédure suivante va nous aidera à simuler notre programme correctement.

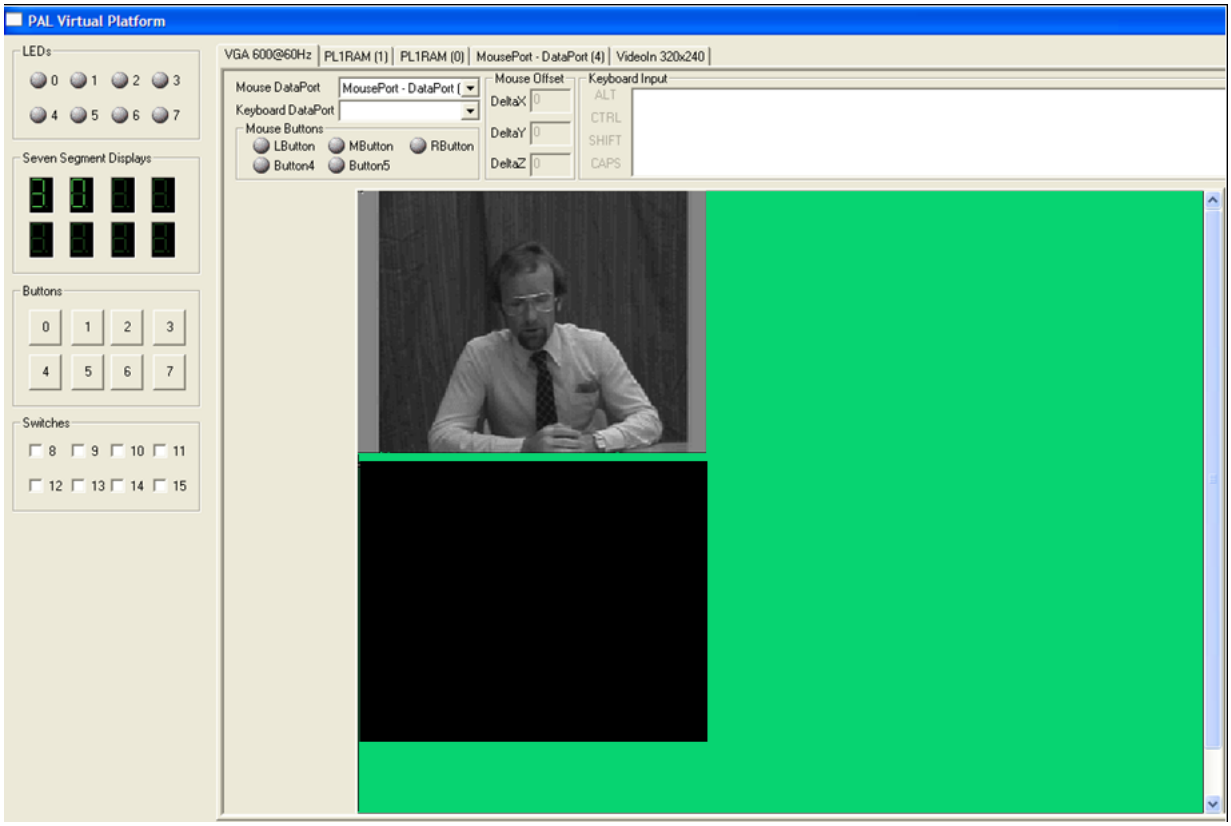


Fig. 5.12 – Début de la simulation

La phase de lecture Pour introduire une image, il faut aller à l'`onglet "VideoIn"` et choisir une image qui joue le role d'une entrée video (`VideoIn`), on revient ensuite à l'`onglet "VGA"`. Pour stocker une image on doit cliquer sur le bouton droit de la souris, l'indice de l'image (*le nombre total d'images stockées*) est affiché sur le premier afficheur sept-segments,

et l'image sera inscrite dans la mémoire `PL1RAM(0)`.

FIG. 5.13 – La mémoire des entrées `PL1RAM(0)`

La phase de traitement Pour calculer les trois filtres, il suffit juste de cliquer sur le bouton gauche de la souris, celui-ci incrémente la valeur de l'image à traiter (*image courante*), l'indice de cette dernière est affiché sur le deuxième afficheur sept-segments, la deuxmième mémoire `PL1RAM(1)` va donc se remplir par les résultats à afficher.

FIG. 5.14 – La mémoire des résultats `PL1RAM(1)`

La phase d'affichage et de choix de la méthode En addition de l'affichage de l'indice du filtre courant sur les deux `LEDs`, on a choisi une couleur du fond de l'écran pour chaque méthode. Initialement rien est affiché, après un clique sur le bouton du mileu de la souris, on bascule d'un filtre à un autre, tout en visualisant le changement du fond ainsi que des `LEDs`.

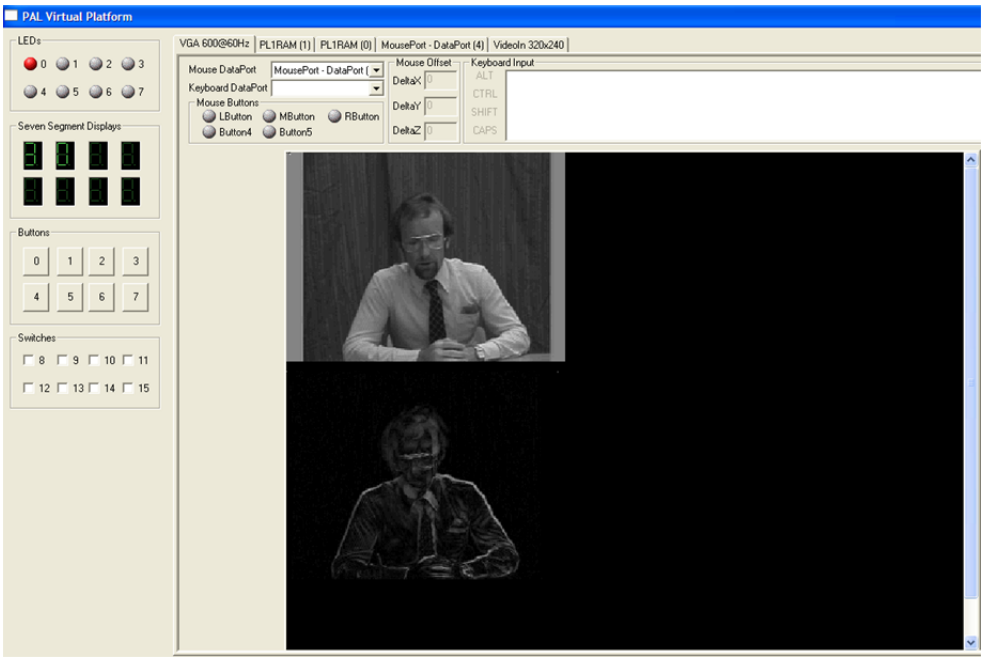


FIG. 5.15 – Affichage du gradient temporel

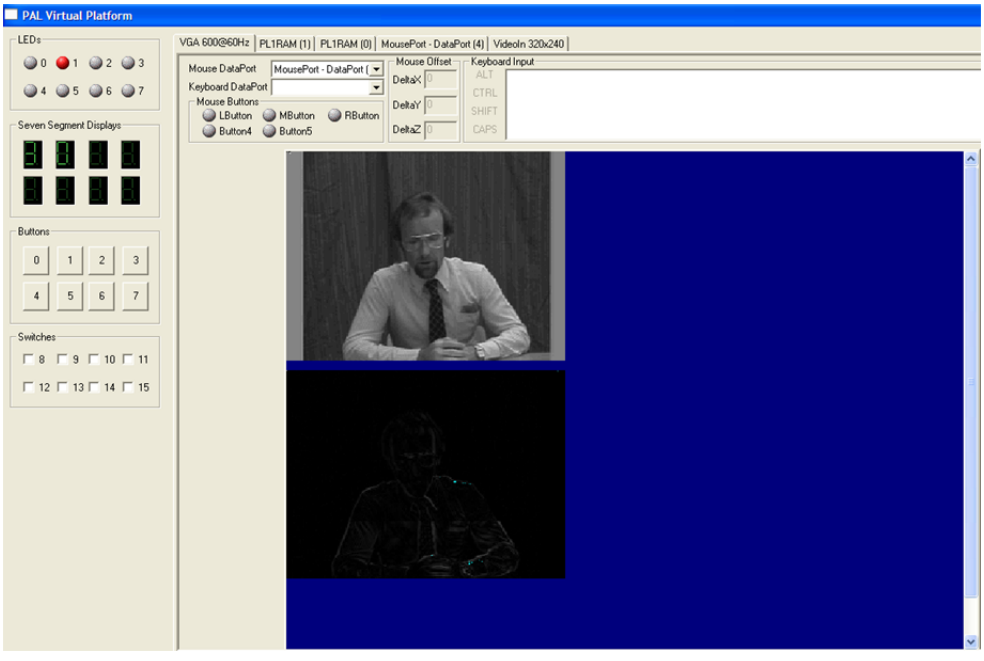


FIG. 5.16 – Affichage de la Moyenne récursive

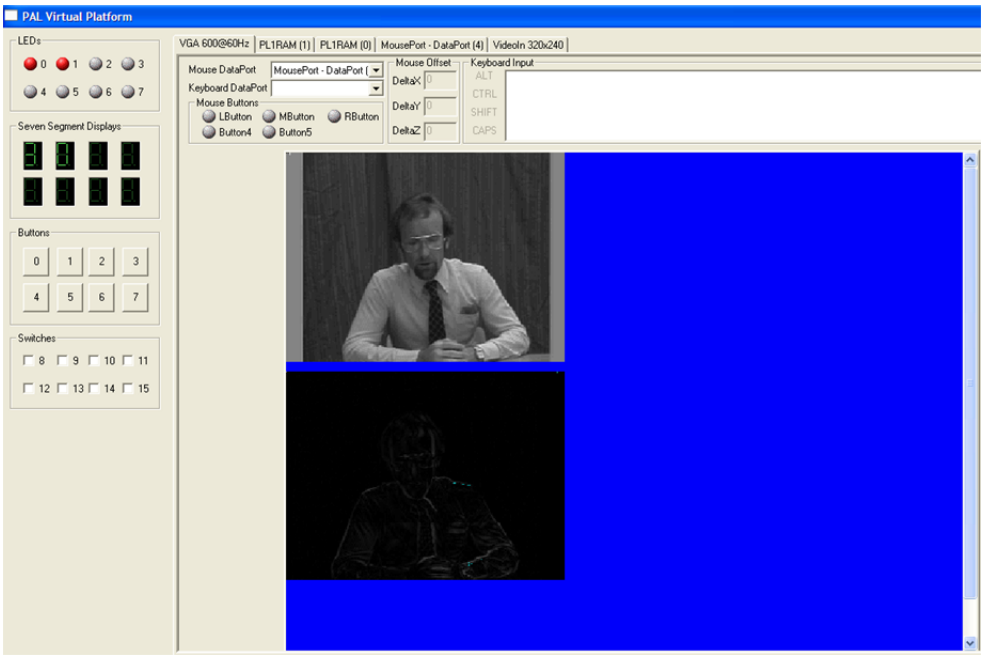


FIG. 5.17 – Affichage du gradient oublieux

5.5 Conclusion

La carte RC203E et son langage de programmation HANDLE-C sont des outils très utiles pour les algorithmes de détection de mouvement, vue que la richesse envisageable de la carte par sa rapidité de calcul et l'espace alloué dans cette carte. le langage présente à son tour une flexibilité dans l'appel des fonctions (VideoIn, VideoOut, RAM, Mouse, ...).

6

Application et présentation des résultats

6.1 Introduction

L'interface MATLAB est un ensemble de programmes affiché graphiquement. Cet interface assure plus de simplicité puisque la manipulation se fait visuellement (avec la souris généralement) loin de l'appel classique des fonctions à partir de l'espace de travail MATLAB qui devient très ennuyeux surtout s'il s'agit d'une fonction usuelle.

6.2 La description de l'interface MATLAB

D'après la figure (Fig - 6.1), cet interface permet donc de charger et visualiser la séquence, choisir le type de traitement et de régler les différents paramètres utilisés.

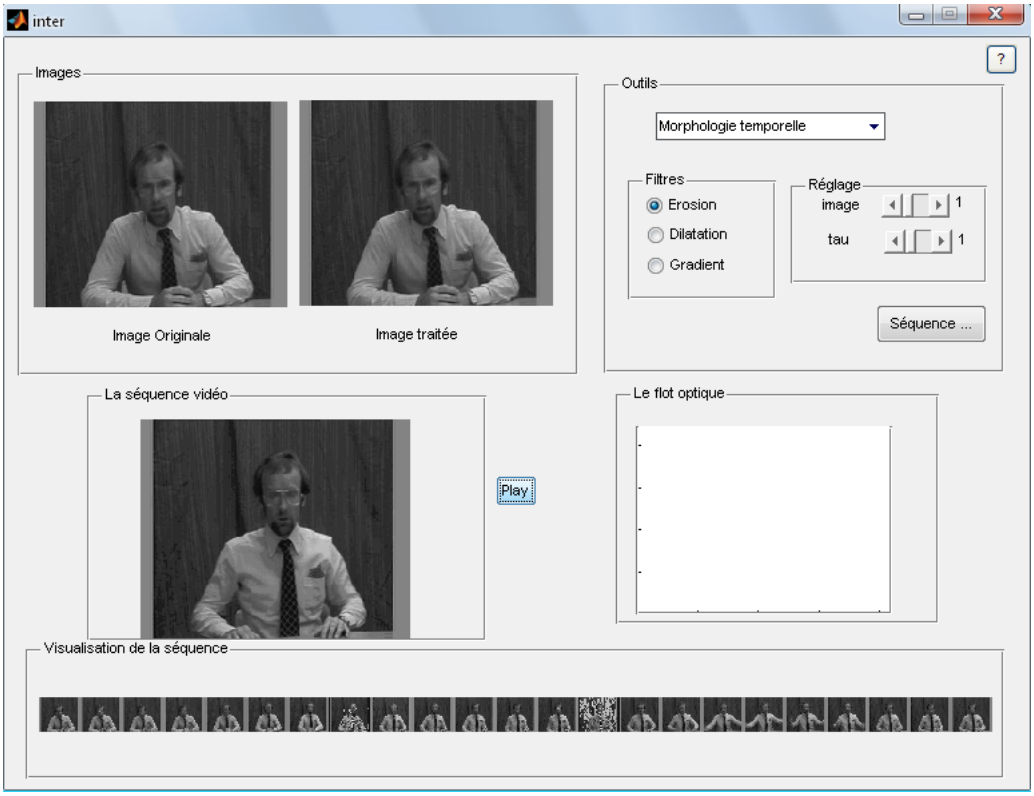


FIG. 6.1 – L'interface MATLAB

6.2.1 Le chargement de la séquence

En appuyant sur le bouton `séquence`, on peut charger la séquence et le nombre d'image qu'on veut sous n'importe quelle format (conversion automatique avec la fonction `readseq` qu'on a mis au point). En allant par exemple dans le répertoire `trevor` et en cliquant sur `trevor23.BMP`. On charge une séquence d'images composée de 23 images qui apparaissent en dessous selon le nombre d'images dans la séquence. Leurs tailles s'adaptent automatiquement avec l'espace de l'interface graphique. Pour visualiser toute la séquence il suffit d'appuyer sur le bouton `Play`.

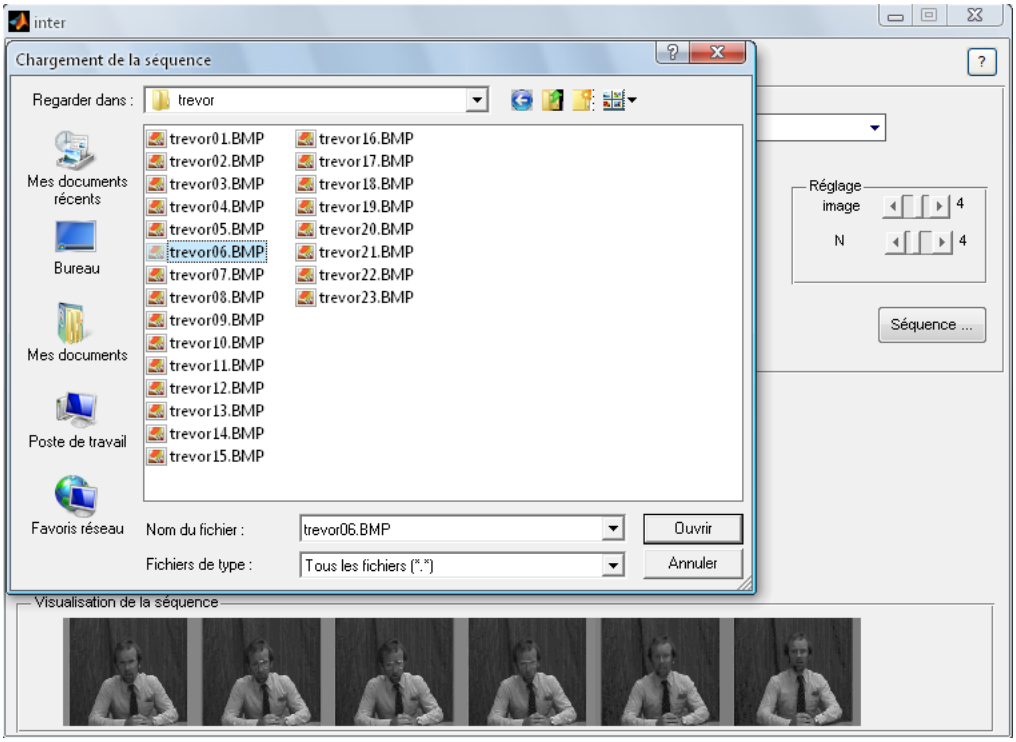


FIG. 6.2 – Le chargement de la séquence

6.2.2 Le choix de la méthode

Une fois la séquence est chargé, il nous reste qu'à choisir la méthode de traitement en utilisant le `combobox` qui est entouré en rouge comme le montre la figure (FIG - 6.3).

A l'exception de la moyenne récursive et le filtre $\Sigma - \Delta$, nous avons plusieurs suppositions et c'est pour cela qu'on a utilisé un `Radiobutton`, la figure (FIG - 6.4) nous montrent qu'on a choisit l'érosion temporelle, et le filtrage moyenneur linéaire.

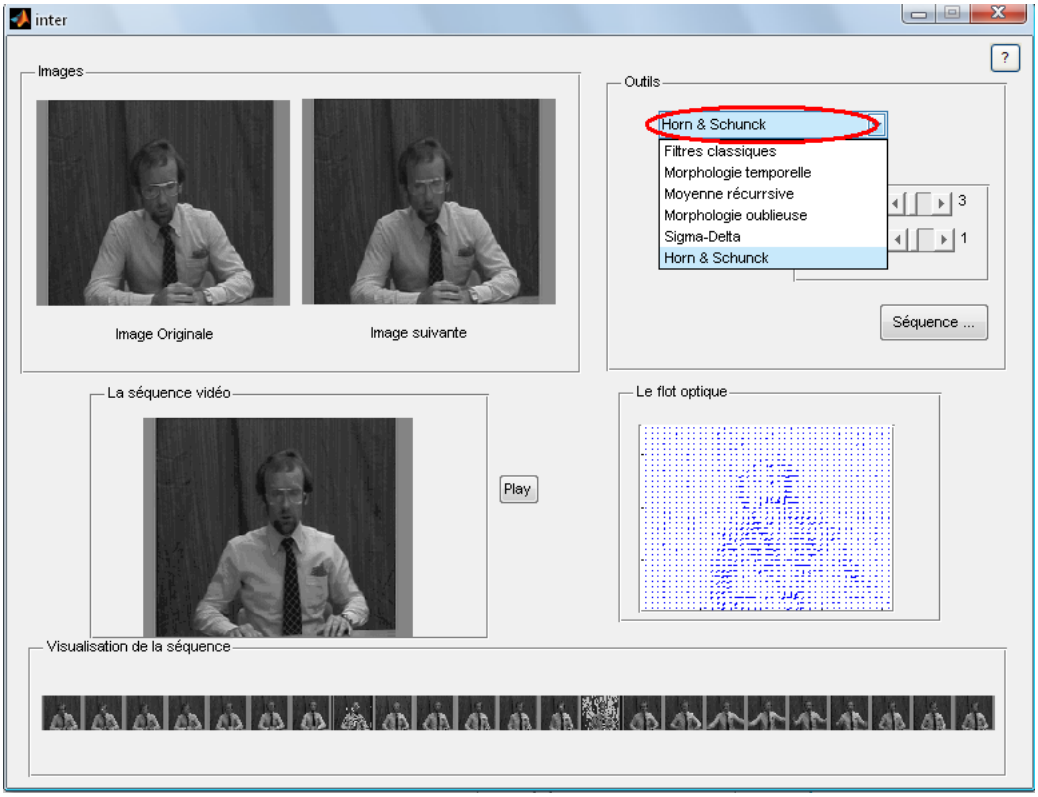


FIG. 6.3 – Le choix de la méthode

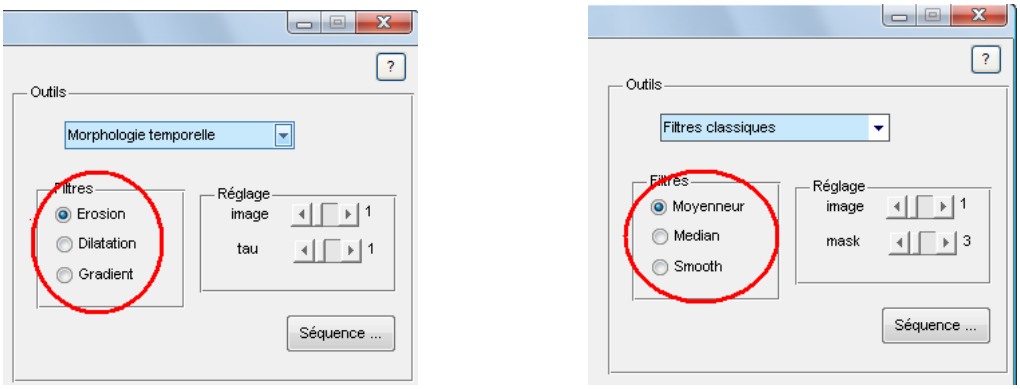


FIG. 6.4 – Les différentes options de chaque méthode

Les figures (FIG - 6.5, FIG - 6.6) présentent les résultats obtenus pour la moyenne récursive en cochant le `checkbox` nous aurons la différence entre l'image et le fond, sinon seulement le fond sera visualisé.

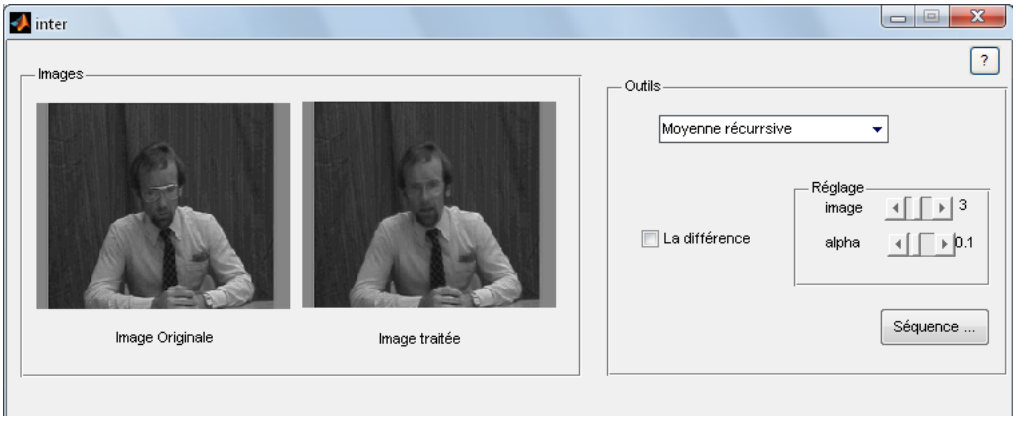


FIG. 6.5 – Le fond de la moyenne récursive

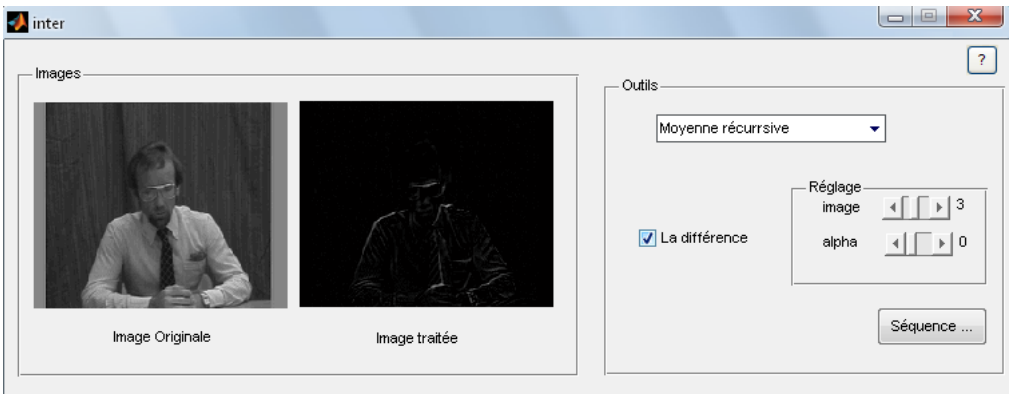


FIG. 6.6 – La différence de la moyenne récursive

6.2.3 Le réglage

En utilisant les `Scrollbar` (FIG - 6.7), on peut choisir l'image désirée ainsi que le paramètre correspondant à chaque technique (le masque pour les filtres linéaires, τ pour la morphologie temporelle, α pour la moyenne récursive, la morphologie oublieuse et N pour le filtre $\Sigma - \Delta$).

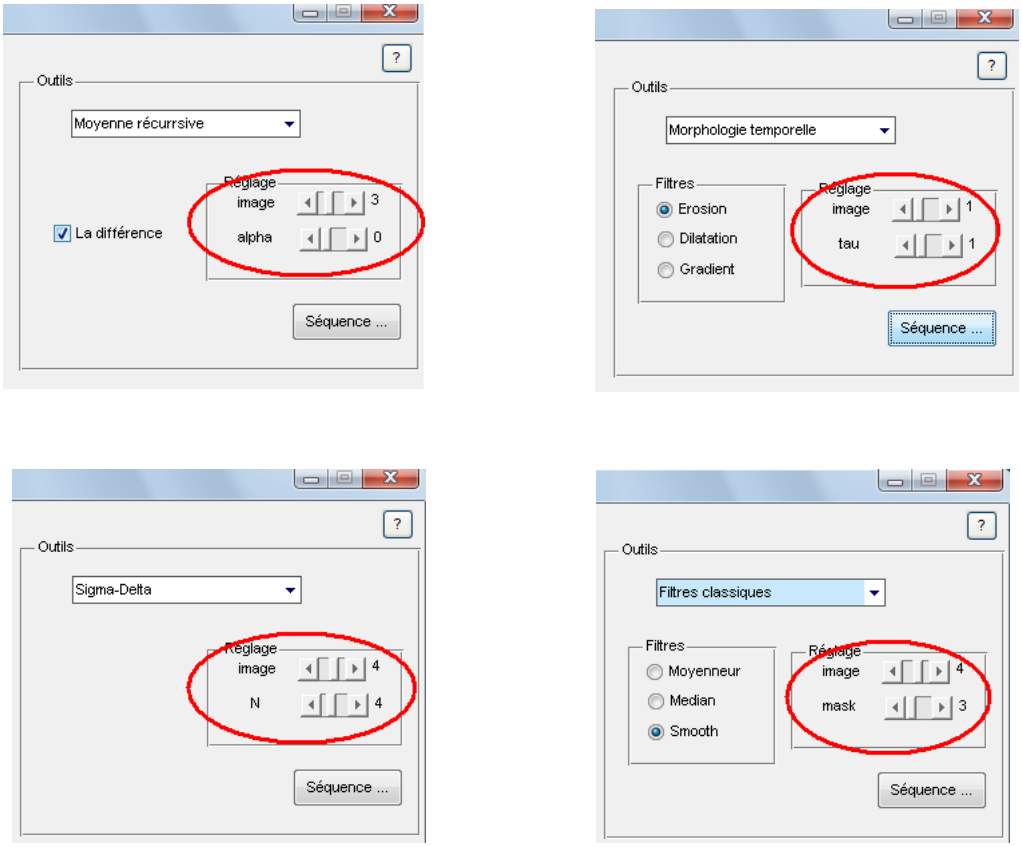


FIG. 6.7 – Le réglage des paramètres

6.3 Résultats et comparaison

En comparant les erreurs de chaque méthode, on constate que chacune d'elle présente des avantages et des inconvénients, pour cela on ne peut pas privilégier une méthode sur une autre sans analyser certains paramètres tels que :

- La nature du bruit qui se présente dans la scène
- Le temps de calcul
- La nature du mouvement et de l'objet

Mais on peut classer ces méthodes par degré de fiabilité suivant ces paramètres

6.3.1 Fiabilité selon la nature du mouvement et de la scène

Plus le mouvement est lent plus nous aurons intérêt à comparer l'image en question avec un maximum d'images qui la suive et/ou la précède, et inversement pour les mouvements rapides. Nous avons choisi la scène `Trevor12-18` (de l'image 12 jusqu'à l'image 18), pour présenter nos résultats dans ce chapitre, parce qu'elle met en évidence ce problème. Ce choix se justifie par la présence du bruit au niveau de la quatorzième image, au cas ou un bruit sera introduit au niveau de l'acquisition, mais surtout parce que cette séquence présente un mouvement lent de l'image 12 jusqu'à l'image 15 et ce mouvement devient rapide de l'image 16 jusqu'à l'image 18.

6.3.2 Fiabilité en filtrage du bruit

C'est à partir du calcul d'erreur (il s'agit de l'EQM/energie totale) qu'on peut discuter la performance en filtrage du bruit. La méthode $\Sigma - \Delta$ nous donne un meilleur filtrage du bruit puisque elle ajuste sa variance avec l'erreur en incrémentant et en décrémentant avec un facteur près N, suivi du gradient oublieux puis la moyenne récursive et enfin le gradient temporel.

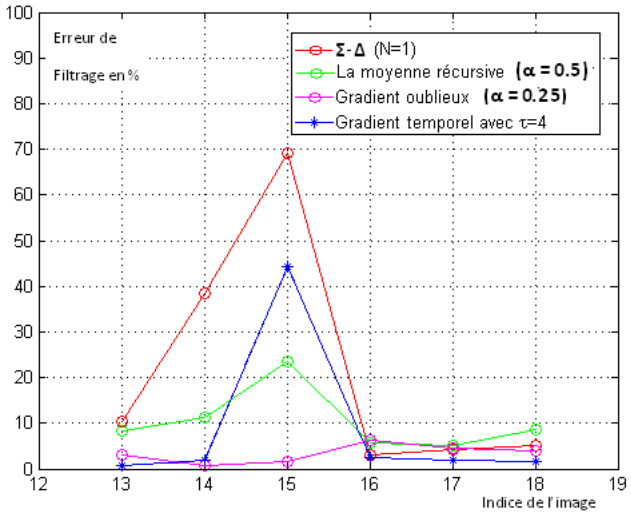


FIG. 6.8 – Performance en filtrage de bruit

Seulement si l'image est trop bruitée, l'estimateur $\Sigma - \Delta$ ne converge plus mais bien au contraire il diverge, c'est le cas de l'image 14 de la séquence. Il devient par conséquent le moins

fiable des quatre méthodes.

6.3.3 Fiabilité en segmentation d'objet

La segmentation d'objet s'interprète par le nombre de point non nuls de chaque image, plus le nombre de ces point est réduit plus nous aurons une meilleure segmentation (les objets sont décrits par un nombre minimum de point).

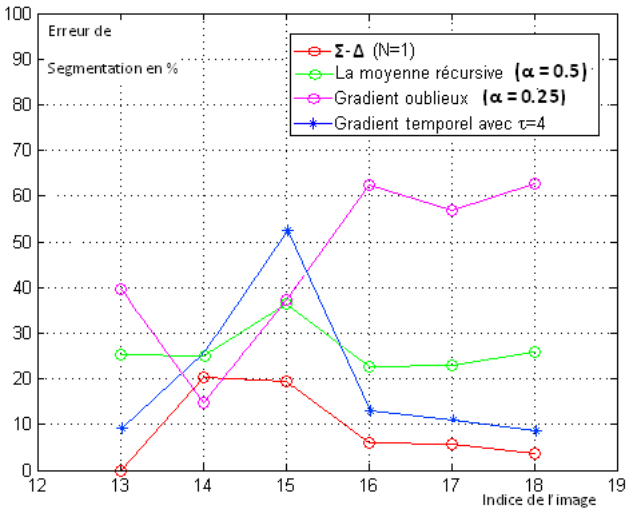


FIG. 6.9 – Performance en segmentation

La figure (FIG - 6.9) nous montre que l'estimateur $\Sigma - \Delta$ est le plus performant en segmentation puis la moyenne récursive et enfin le gradient oublieux.

6.3.4 Fiabilité en temps de calcul

Le temps de traitement varie d'une méthode à une autre lors de l'implémentation sur le circuit FPGA, on peut classer le gradient temporel comme étant le plus rapide puisqu'il s'agit d'une simple comparaison entre les différents trames d'image stockées en mémoire (recherche des maxima et des minima). vient en second lieu la moyenne récursive, où on fait appel à une simple multiplication (décalage). Quant au gradient oublieux il occupe la troisième place à cause de son mode de calcul récursif qui fait appel non seulement à une multiplication mais aussi à une comparaison, et enfin l'estimateur $\Sigma - \Delta$ qui utilise une multiplication, deux

comparaisons, des additions et des incrémentations/décrémentations.

6.3.5 Etude de l'estimateur $\Sigma - \Delta$

Dans cette partie, nous comptons étudier l'estimateur $\Sigma - \Delta$ de plus près, en calculant l'erreur qui interprète la performance en filtrage du bruit et le nombre de points non nuls qui interprètent la performance en segmentation. En variant à chaque fois le paramètre de convergence N.

images	Erreur de filtrage en %	erreur de segmentation en %
12-13	9.70	0
13-14	39	20.28
14-15	**69.5**	**19.23**
15-16	3.16	5.98
16-17	6.16	5.59
17-18	4.71	3.57

TAB. 6.1 – Les erreurs de filtrage et segmentation de l'estimateur $\Sigma - \Delta$ avec N=4.

images	Erreur de filtrage en %	erreur de segmentation en %
12-13	10.44	0
13-14	38.43	20.43
14-15	**69.16**	**19.37**
15-16	2.98	6.10
16-17	4.29	5.59
17-18	4.98	3.69

TAB. 6.2 – Les erreurs de filtrage et segmentation de l'estimateur $\Sigma - \Delta$ avec N=1.

On remarque bien que dans les figures suivantes, les trois courbes sont très proches l'une de l'autre, on en déduit donc que N ne réduit ni l'erreur de filtrage ni l'erreur de segmentation, mais il agit seulement sur la rapidité de convergence.

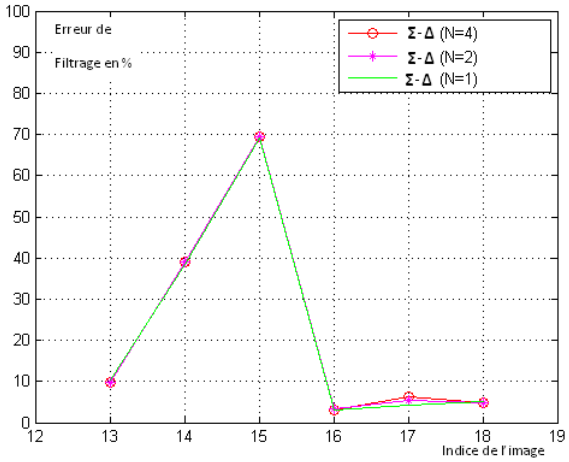


FIG. 6.10 – Le filtrage du bruit par l'estimateur $\Sigma - \Delta$

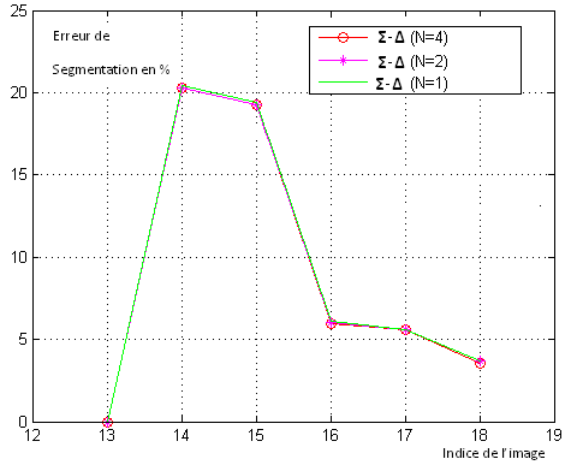


FIG. 6.11 – L'erreur de segmentation de l'estimateur $\Sigma - \Delta$

6.3.6 Etude de la moyenne récursive

Les tableaux (TAB - 6.3, TAB - 6.4) présentent les résultats obtenus pour $\alpha = 0.5$ et $\alpha = 0.25$.

On remarque que la moyenne récursive avec $\alpha = 0.5$ filtre mieux le bruit qu'avec un $\alpha = 0.25$, ceci est dû au changement de luminance de la scène car elle était sombre au début puis elle devient de plus en plus éclairée, dans ce genre de situation il vaux mieux que notre α soit plus proche de 1 et inversement. (c'est à dire lorsque la scène claire au début devient sombre il est recommandé de choisir un α proche de 0), c'est pour cela qu'on envisage deux

images	Erreur en %	Segmentation en %
12-13	0.71	8.31
13-14	2.04	24.87
14-15	**44.43**	**51.72**
15-16	2.58	12.12
16-17	2.00	10.10
17-18	1.60	7.62

TAB. 6.3 – L'erreur de la moyenne récursive avec $\alpha = 0.5$.

images	Erreur en %	Segmentation en %
12-13	4.85	25.43
13-14	2.79	25.06
14-15	**59.91**	**36.45**
15-16	3.57	22.68
16-17	5.40	22.84
17-18	8.09	25.82

TAB. 6.4 – L'erreur de la moyenne récursive avec $\alpha = 0.25$.

zones dans le graphe de la segmentation.

- Zone 1 (12-15 La scène sombre) l'image est mieux segmenté pour $\alpha = 0.5$ que pour $\alpha = 0.25$.
- Zone 2 (15-18 La scène claire) l'image est mieux segmenté pour $\alpha = 0.25$ que pour $\alpha = 0.5$.

En conclusion la moyenne récursive est sensible au changement de la luminance, donc il faut connaître la scène au préalable pour un avoir le bon α.

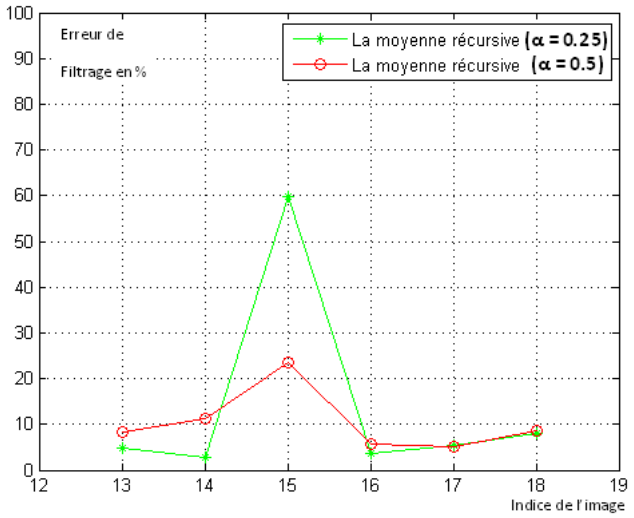


FIG. 6.12 – l'erreur de filtrage de la moyenne récursive

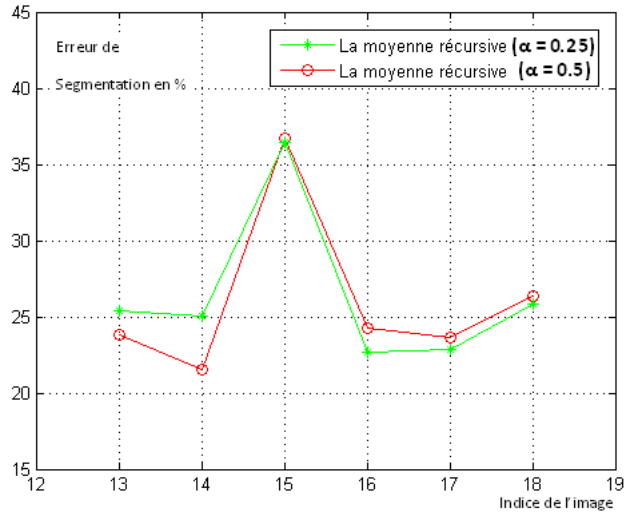


FIG. 6.13 – l'erreur de segmentation de la moyenne récursive

6.3.7 Etude du gradient temporel

Dans cette partie on prend un intervalle $\tau = [0, 4]$. Les résultats obtenus sont illustrés dans le tableau (TAB - 6.5).

images	Erreur de filtrage en %	erreur de segmentation en %
12-13	0.71	8.31
13-14	2.04	24.87
14-15	**44.43**	**51.72**
15-16	2.58	12.12
16-17	2.00	10.10
17-18	1.60	7.62

TAB. 6.5 – Les erreurs du gradient temporel avec $\tau = [0, 4]$

6.3.8 Etude du gradient oublieux

L'avantage du gradient oublieux c'est qu'il annule complètement le bruit, c'est le cas de l'image 14 (FIG - 6.14), mais il est moins performant en segmentation comparativement au gradient temporel.

images	Erreur en %	Segmentation en %
12-13	5.03	39.74
13-14	0.52	14.89
14-15	**1.66**	**37.21**
15-16	9.92	62.29
16-17	5.03	56.78
17-18	5.81	62.66

TAB. 6.6 – L'erreur du gradient oublieux avec $\alpha = 0.5$.

images	Erreur en %	Segmentation en %
12-13	3.04	72.62
13-14	0.77	29.91
14-15	**1.68**	**43.26**
15-16	6.29	65.24
16-17	4.66	61.88
17-18	4.01	66.95

TAB. 6.7 – L'erreur du gradient oublieux avec $\alpha = 0.25$.

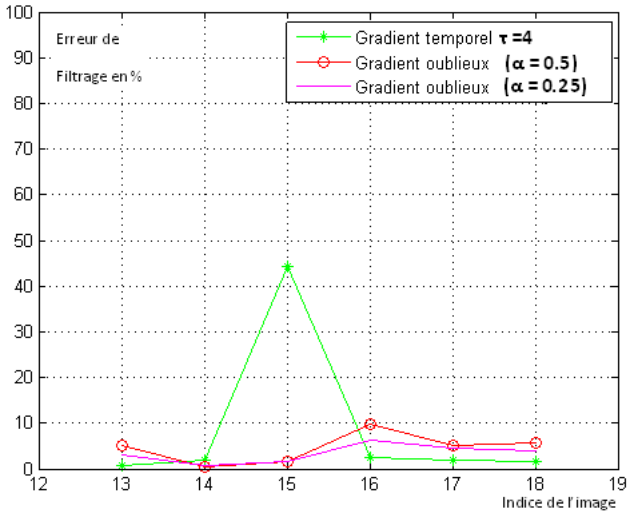


FIG. 6.14 – L'erreur de filtrage des gradients (temporel et oublieux)

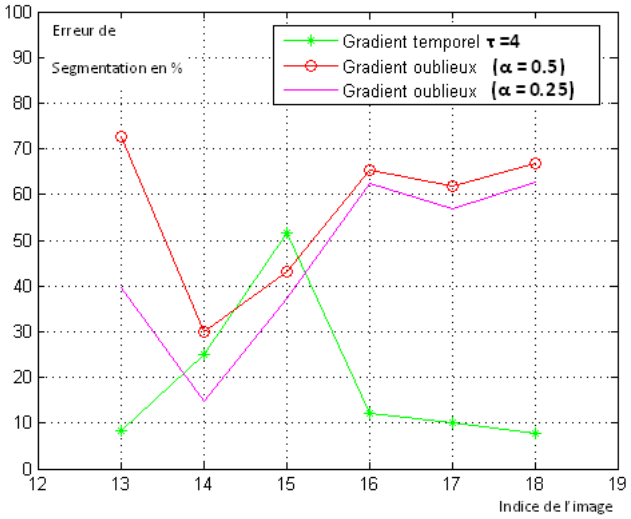


FIG. 6.15 – L'erreur de segmentation des gradients(temporel et oublieux)

On remarque que le pic de l'image 14 apparaît très clairement pour le gradient temporel, c'est pour cette raison qu'on dit que le gradient temporel maximise les erreurs. Avec un intervalle assez grand $\tau = [0, 4]$, on constate que l'erreur de segmentation diminue est nous donne des résultats meilleurs à ceux apportés par le gradient oublieux.

6.4 Conclusion

D'une manière générale on peut classer les performances des différentes méthodes étudiées par ordre décroissant, pour les critères suivants :

Filtrage

- $\Sigma - \Delta$ à moins que l'image n'est pas beaucoup bruitée.
- Le gradient oublieux qui est indépendant aux paramètres de la scène (luminance et bruit).
- La moyenne récursive qui dépend aussi fortement de la luminance.
- Le gradient temporel qui maximise le bruit.

Segmentation

- $\Sigma - \Delta$.
- La moyenne récursive.
- Le gradient temporelle avec un intervalle assez grand pour les mouvement lents sinon il est le moins fiable des quatre.
- Le gradient oublieux.

Conclusion Générale

Dans ce mémoire, nous avons étudié et implémenté quatre méthodes différentes pour la détection du mouvement dans une séquence d'images. Ces méthodes sont capables de filtrer l'image au préalable pour obtenir une meilleure estimation du mouvement réel et résoudre aussi le problème de la variation de la luminance posé par les approches classiques (différentielles, corrélation de blocs, ...).

Les algorithmes choisis sont le gradient temporel, le gradient oublieux, la moyenne récursive et l'estimateur $\Sigma - \Delta$. Cette dernière est considérée comme la plus performante et la plus robuste selon les critères définis auparavant (sensibilité aux bruits, segmentation de la région en mouvement, rapidité).

Nous avons par la suite proposé une architecture hardware basée sur la carte RC203E de Celoxica et le circuit FPGA Xilinx Virtex II. Cette architecture est obtenue en utilisant Handel-C et englobe trois algorithmes (gradient temporel, gradient oublieux et moyenne récursive), afin d'accélérer le temps de calcul.

Enfin, nous souhaiterons que notre travail puisse contribué à la réalisation d'un système en temps réel pour la télésurveillance ou la détection d'obstacle en robotique mobile.

Notre projet, nous a permis aussi de combiner l'informatique et l'électronique pour concevoir un système plus pratique et plus performant.

A

Scripts MATLAB utilisés

A.1 Le filtre moyenneur

```
function[S]=moyenno(A,m)
n=size(A) ;
l=fix(m/2) ;
S=A ;
for i=l+1 :n(1)-l
for j=l+1 :n(2)-l
S(i+l,j+l)= sum(sum(A(i :i+m-1,j :j+m-1)))/(m*m)
end
end
```

A.2 Le filtre médian

```
function[S]=mediano(A,m)
n=size(A) ;
l=fix(m/2) ;
S=A ;
for i=l+1 :n(1)-l
for j=l+1 :n(2)-l
C=0 ;
for k=i-l :i+l
C=[C (A(k,j-l :j+l))] ;
```

```
end
C(1)=[ ] ;
S(i,j)=median(C) ;
end
end
```

A.3 Le filtre Smooth

```
function[S]=smth(A,m)
n=size(A) ;
l=fix(m/2) ;
S=A ;
i=-l :l ;
hx=exp(-i.*i/2) ;
B=hx'*hx ;
SM=ceil(B/min(min(B))) ;
SM=SM/sum(sum(SM)) ;
C=zeros(m,m) ;
for i=l+1 :n(1)-l
for j=l+1 :n(2)-l
for k=-l :l
for p=-l :l
C(p+l+1,k+l+1)=A(i+p,j+k)*SM(p+l+1,k+l+1) ;
end
end
S(i,j)=sum(sum(C)) ;
end
end
```

A.4 La corrélation de phase

```
function [F]=corr(I1,I2)
F1=fft2(I1) ;
F2=fft2(I2) ;
SPCN=F2.*conj(F1) ;
spcn=SPCN/(max(max(abs(SPCN)))) ;
x=abs(ifft2(spcn)) ;
F=fix(255*x/max(max(x))) ;
```

A.5 Le script de la dérivation

```
function [S]=dru(I1,I2,para)
n=size(I1);
if para==1
S=[zeros(n(1),1) I1( :,1 :n(2)-1)]-I1;
elseif para==2
S=[zeros(1,n(2)); I1(1 :n(1)-1, :)]-I1;
else
S=I2-I1;
end
```

A.6 L'algorithme de Horn et Schunck

```
function [vx,vy]=flot(I1,I2,alpha)
Ix=dru(I1,I2,1);
Iy=dru(I1,I2,2);
It=dru(I1,I2,0);
n=size(It);
vxk=0*ones(n(1),n(2));
vyk=0*ones(n(1),n(2));
a=alpha*ones(n(1),n(2));
for k=0 :20
vx=vxk-(double(Ix).*(double(Ix).*vxk+double(Iy).*vyk+...
double(It)))./(a+double(Ix).*double(Ix)+double(Iy).*double(Iy));
vy=vyk-(double(Iy).*(double(Ix).*vxk+double(Iy).*vyk+...
double(It)))./(a+double(Ix).*double(Ix)+double(Iy).*double(Iy));
vxk=vx;vyk=vy;
end
```

A.7 L'acquisition de la séquence

```
function [Seq]=readseq(path,nom,type,count)
a=' ';
a=sprintf('% s% s01.% s',path,nom,type);
[A tab]=imread(a);
Seq=A;
n=size(A);
for m=1 :n(1)
for k=1 :n(2)
Seq(m,k)=fix(tab(A(m,k)+1,1)*255);
```

```
end
end
for i=2 :count
if(i<10)
a=sprintf('% s% s0% d.% s',path,nom,i,type);
else
a=sprintf('% s% s% d.% s',path,nom,i,type);
end
[A1 tab1]=imread(a);
A2=A1;
for m=1 :n(1)
for k=1 :n(2)
A2(m,k)=fix(tab(A1(m,k)+1,1)*255);
end
end
Seq=[Seq A2];
end
```

A.8 L'acquisition de l'image

```
function [img]=getim(Seq,idx,count)
n=size(Seq);
n2=n(2)/count;
img=zeros(n(1),n2);
img=Seq( :,(idx-1)*n2+1 :idx*n2);
```

A.9 La dilatation temporelle

```
function [img]=dilt(Seq,capt,tau,count)
if capt+tau<count
tau=count-capt;
end
img=getim(Seq,capt,count);
for i=capt :capt+tau
img=max(getim(Seq,i,count),img);
end
```

A.10 L'érosion temporelle

```
function [img]=erot(Seq,capt,tau,count)
if capt+tau>count
tau=count-capt ;
end
img=getim(Seq,capt,count) ;
for i=capt :capt+tau
img=min(getim(Seq,i,count),img) ;
end
```

A.11 Le gradient temporel

```
function [img]=gradt(Seq,capt,tau,count)
img=dilt(Seq,capt,tau,count)-erot(Seq,capt,tau,count) ;
end
```

A.12 La dilatation oublieuse

```
function [img]=oubdilt(Seq,idx,alpha,count)
M0=getim(Seq,1,count) ;
img=M0 ;
for i=2 :idx
It=getim(Seq,i,count) ;
img=alpha*It+(1-alpha)*max(It,M0) ;
M0=img ;
end
```

A.13 L'érosion oublieuse

```
function [img]=oubero(Seq,idx,alpha,count)
M0=getim(Seq,1,count) ;
img=M0 ;
for i=2 :idx
It=getim(Seq,i,count) ;
img=alpha*It+(1-alpha)*min(It,M0) ;
M0=img ;
end
```

A.14 Le gradient oublieux

```
function [img]=oubgrad(Seq,idx,alpha,count)
img=oubdilt(Seq,idx,alpha,count)-oubero(Seq,idx,alpha,count) ;
```

A.15 La moyenne recusive

```
function [img]=moyrecc(Seq,alpha,idx,count)
img=getim(Seq,1,count) ;
for i=1 :idx
S=alpha*getim(Seq,i,count)+(1-alpha)*img ;
img=S ;
end
```

A.16 L'estimateur $\Sigma - \Delta$

```
function [D]=moysd(Seq,idx,count)
MO=getim(Seq,1,count) ;
for k=2 :idx
Mt=MO-sign(getim(Seq,k,count)-MO) ;
MO=Mt ;
end
D=Mt-getim(Seq,idx,count) ;
```

```
function [D]=varsd(Seq,idx,N,count)
VO=moysd(Seq,2,count) ;
n=size(VO) ;
Vt=zeros(n(1),n(2)) ;
for k=2 :idx
A=moysd(Seq,k,count) ;
for i=1 :n(1)
for j=1 :n(2)
if A(i,j) =0
Vt(i,j)=VO(i,j)+sign(N*abs(A(i,j))-VO(i,j)) ;
end
end
end
VO=Vt ;
end
D=Vt ;
```

```
function [D]=sd(Seq,idx,N,count)
A=double(moysd(Seq,idx,count))-double(varsd(Seq,idx,N,count)) ;
n=size(A) ;
D=zeros(n(1),n(2)) ;
for i=1 :n(1)
for j=1 :n(2)
if sign(A(i,j))  =0
D(i,j)=(sign(A(i,j))+1)/2 ;
end
end
end
```

Bibliographie

[BS03] BS. *Platform Developer's Kit (RC200/203 hardware and PSL Reference Manual).* Celoxica ,http ://www.celoxica.com/support/, 2003.

[CCB01] Daniel COLLOBERT, Michel COLLOBERT, Olivier BERNIER. *Détection stochastique du mouvement dans les séquences vidéos (Application à la compression).* France Telecom R&D, 2001.

[GRA03] Cristian GRAVA. *Compensation de mouvement par réseaux neuronaux cellulaires. Application en imagerie médicale.* Thèse Institut National des Sciences Appliquées de Lyon, 2003.

[HS81] B.K.P. HORN et B.G. SCHUNCK. *Determining Optical Flow.* Massachusetts Institute of Technology, Cambridge, 1981.

[MAN05a] Antoine MANZANERA. *Cours de morphologie mathématique.* ENSTA/ LEI UPMC/ MASTER IAD, 2005.

[MAN05b] Antoine MANZANERA. *Techniques du traitement d'images (Estimation du mouvement dans les séquences d'images).* Master IA D ENSTA/LEI, 2005.

[MR04a] A.MANZANERA et J.C.RICHEFEU. *A new hybrid differential filter for motion detection.* ENSTA/ UEI, 2004.

[MR04b] A.MANZANERA et J.C.RICHEFEU. *A robust and computationally efficient motion detection algorithm based on $\Sigma-\Delta$ background estimation.* Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'04). Kolkata - India, 2004.

[MR07] A.MANZANERA et J.C.RICHEFEU. *A new motion detection algorithm based on* $\Sigma - \Delta$ *background estimation.* Progress in Pattern Recognition, Image Analysis and Applications (CIARP'07). Viña del Mar-Valparaïso, Chile, 2007.

[PAI01] Michel PAINDAVOINE. *Traitement des images en temps réel.* Techniques de l'ingénieur, 2001.

[PC95] S. PHILIPP J. COCQUEREZ. *Analyse d'images : filtrage et segmentation.* Edition MASSON, 1995.

[RIC06] Julien RICHEFEU. *Détection et analyse du mouvement sur système de vision à base de rétine numérique.* UNIVERSITÉ PARIS 6, 2006.

[RT06] Nadjah ROULA et Sabrina TIRES. *Implémentation d'algorithmes de traitement d'images sur DSP C6000 (Application à la séparation des Chromosomes).* PFE Ecole Nationale Polytechnique, 2006.

[SS01] F.SABEH et H.SAHRAOUI. *Estimation du mouvement apparent 2D dans une séquence d'images par mise en correspondance.* Projet fin d'études, INi, 2001.

[VER06] François VERDIER. *Les circuits FPGA Concepts de base, architecture et applications.* Université de Cergy-Pontoise/Laboratoire ETIS, 2006.