République **A**lgérienne **D**émocratique et **P**opulaire

Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

Université des **S**ciences et de la **T**echnologie d'**O**ran **M**ohamed **B**oudiaf

Faculté de Génie Electrique **D**épartement d'Electronique

THÈSE

En vue de l'obtention du

Diplôme de Doctorat

Présentée et Soutenue par :

KERFA Djoudi

Intitulée

Estimation de mouvement d'objets dans une séquence d'images

Domaine *: Sciences et Technologies*
Spécialité *: Electronique*
Intitulé de la Formation *: Signaux et Systèmes Intelligents*

Le jury est composé de :

Grade,	***Nom &Prénom***	***Statut***	***Domiciliation***
Professeur,	*BOUGHANMI. Nabil*	*Président*	*USTO-MB*
Professeur,	*BELBACHIR. M. Faouzi*	*Directeur de thèse*	*USTO-MB*
Professeur,	*DJEBBARI. Ali*	*Examinateur*	*Univ, S.B.A*
Professeur,	*FEHAM. Mohammed*	*Examinateur*	*Univ, Tlemcen*
Professeur,	*DEROUICHE. Ziane*	*Examinateur*	*USTO-MB*
M.C. A,	*BENMOUSSAT. Nawal*	*Examinatrice*	*USTO-MB*

Année Universitaire 2015 / 2016

REMERCIEMENTS

Je remercie ALLAH, *le tout puissant, pour m'avoir donné, le courage, la patience, la volonté et la force nécessaire, pour l'accomplissement de ce travail.*

Je tiens à exprimer toute ma gratitude à mon Directeur de thèse, Mr. Mohamed Faouzi Bel bachir pour m'avoir guidé pendant ces années de thèse. Je n'aurais jamais pu mener ce travail à bien sans son exigence, sa disponibilité et ses conseils pertinents. Ses grandes qualités scientifiques et humaines m'ont aidé, motivé et orienté tout en m'offrant une grande liberté dans la recherche. J'ai particulièrement apprécié l'autonomie qu'il m'a accordé dans les choix et les orientations de mon travail. Je n'oublierai jamais ses encouragements tant au niveau personnel que professionnel. J'en suis vraiment très reconnaissant.

Je tiens à exprimer mes plus sincères remerciements à tous ces professeurs ; Mr.Nabil Boughanmi qui m'a fait un grand honneur en acceptant de présider le jury de cette thèse. J'exprime toute ma gratitude à Mr.Ali Djebbari, Mr.Mohammed Feham, à Mme. Nawal Benmoussat et Mr.Ziane Derouiche pour avoir accepté d'être les examinateurs de cette thèse.

Je remercie très chaleureusement les membres de notre laboratoire LSSD pour le soutien et les conversations scientifiques qu'on a eu, ils m'ont souvent permis de voir mon travail sous un autre angle et de l'améliorer.

Enfin les mots sont trop pauvres pour exprimer ma gratitude à mon père, à ma mère, à mes sœurs, frères, amis, et à tous ceux qui m'ont soutenu de prés ou de loin à réussir ce travail.

RÉSUMÉ

Cette thèse est dédiée à l'estimation du mouvement dans une séquence d'images. Un état de l'art est présenté montrant que les algorithmes BMA (Block Matching Algorithm) sont les plus utilisés pour leur efficacité. C'est ce qui explique l'intérêt donné dans cette thèse à ce type d'algorithmes.

Nous proposons trois nouveaux algorithmes. Le premier algorithme nommé par Étoile Diamant ED effectue la recherche du vecteur de mouvement en 2 étapes. Après une recherche grossière (Schéma Étoile) une recherche plus précise est réalisée grâce au schéma Petit Diamant. Cet algorithme ED a été conçu en prenant en considération une étude statistique que nous avons effectuée sur un corpus de 12 séquences d'images. Le deuxième algorithme que nous avons nommé par ED_S utilise un seuil S qui permet de réduire encore plus la complexité pour la recherche du vecteur mouvement. ED_S effectue la recherche que si l'erreur centrale dépasse le seuil S. La troisième contribution porte sur l'élaboration d'une méthode simple et efficace pour la détection d'objets mobiles.

Pour toutes les méthodes que nous avons élaborées une étude sur de nombreux exemples a été effectuée montrant leur intérêt ainsi que leur efficacité.

Mots clés :*Estimation et compensation de mouvement, Mise en correspondance de blocs, Codage et compression vidéo, H264/AVC, Détection d'objets en mouvement.*

ABREVIATIONS

AVC : Advanced **V**ideo **C**oding

BMA : Block **M**atching **A**lgorithm

CABAC : Context **A**daptive **B**inary **A**rithmetic **C**oding

CAVALC : Context **A**daptive **V**ariable **L**ength **C**oding

Cb : Chrominance **b**leu

Cr : Chrominance **r**ouge

CIF : Common **I**ntermediate **F**ormat

CS : Cross **S**earch

CDS : Cross **D**iamond **S**earch

CSF : Contrast **S**ensitivity **F**unction

CM : Contraste de **M**astquage

CSP : Cross **S**earch **P**attern

DOM-CHM : Détection d'**O**bjet **M**obile avec **CH**amps de **M**ouvement

DCT ou TCD : Discrete **C**osine **T**ransform,**T**ransformée en **C**osinus **D**iscrète

DS : Diamond **S**earch

DVD : Digital **V**ideo **D**isk

DPN : Data **P**rocess **N**etworks

dpi : dots **p**er **i**nch

ED : Etoile **D**iamant

EM : Estimation de **M**ouvement

ES : Elément **S**tructurant

FS : Full **S**earch

FVM : Frequence des **V**ectreurs de **M**ouvement

FSS ou 4SS : Four **S**tep **S**earch

Gbits/s : Gigabits par **s**econde

Go : Giga **o**ctet

GOP : Group **O**f **P**ictures

HDTV : High **D**efinition **T**ele**V**ision

HDS : Hexagon **D**iamond **S**earch

HEVC : High **E**fficiency **V**ideo **C**oding

HEX : Hexagon **S**earch

IP : Internet **P**rotocol

ISDN : Integrated **S**ervices for **D**igital **N**etwork.

ISO−IEC : International **O**rganization for **S**tandardization−**I**nternational **E**lectrotechnical **C**ommission

JPEG : Joint **P**hotographic **E**xperts **G**roupe

LHSP : Large **H**exagon **S**earch **P**attern

LDSP : Large **D**iamond **S**earch **P**attern

Mvt : Mouvement

MoG : Mixture **o**f **G**aussian

MAD : Mean **A**bsolute **D**ifference

MB : Macro **B**loc

MPEG : Moving **P**icture **E**xperts **G**roup

MSE : Mean **S**quare **E**rror

Mo : Megaoctets

NPC : Nombre de **P**oints **C**alculés

nb : nombre de **b**locs

NTSS : New **T**hree **S**tep **S**earch

NOCDS : Novel **O**ctogon **C**ross **D**iamond **S**earch

PD : Petit **D**iamant

PS : Peack **S**ignal

PSNR : Peack **S**ignal-to-**N**oise **R**atio

4G : Quatrieme **G**eneration

RVB : Rouge **V**ert **B**leu

VM : Vectreurs de **M**ouvement

RNIS : Réseau **N**umérique **I**ntégration de **S**ervice

S : Seuil

SATD : Sum of **A**bsolute **T**ransformer **D**ifferences (Sum of Absolute Hadamard Transformed Differences)

SAD : Sum of **A**bsolute **D**ifferences

SES : Simple **E**fficace **S**earch

SSIM : Structural **S**imilarity **I**ndex **M**easurement

TSS : Three **S**tep **S**earch

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

Contexte général

L'évolution exponentielle des techniques de compression et de codage vidéo, accompagnées par la diversification des appareils capables de lire ces contenus (smartphones, tablettes, ordinateurs personnels, télévisions haute définition, etc) ainsi que la multiplication des supports de transmissions (réseaux wi-fi, réseaux cellulaires, fibre optique, 3G, 4G, etc....) nécessite aux fournisseurs d'investir pour être en mesure de fournir une qualité de vidéo optimale quel que soit le contexte.

L'exigence de la compression des médias numériques et de la vidéo en particulier est évidente. Elle représente ainsi un levier majeur pour atteindre de nombreux objectifs. En effet, la compression ou codage de la vidéo est le procédé permettant de réduire la quantité de données à transmettre ou à stocker, et donc de réduire les coûts qui y sont liés, ou de lever des verrous technologiques en apportant le signal vidéo là où on ne pouvait le transmettre auparavant, ou encore à un plus grand nombre de personnes simultanément.

L'amélioration du taux de compression est un sujet de recherche crucial qui anime une forte communauté de laboratoires académiques et de compagnies internationales. Plusieurs standards de compression ont ainsi été proposés afin d'aboutir à l'amélioration de l'efficacité de codage. Ainsi, MPEG−4, H.264/AVC permet de diminuer le débit nécessaire (jusqu'à 50%) par rapport à son prédécesseur MPEG−2, ouvrant la voie à de nouveaux services, tels que la vidéo haute définition[1].

Un des éléments clé des standards actuels de compression vidéo demeure l'estimation du mouvement. Pendant cette phase, l'image courante est fractionnée en blocs. Pour chaque bloc on cherche le meilleur appariement entre deux images successives. C'est l'opération de mise en correspondance, dont on déduit, pour chaque bloc, un vecteur mouvement. Cette opération participe pour une grande partie à l'efficacité de la compression en permettant de supprimer les redondances temporelles. Pour autant la méthode de détermination des vecteurs de mouvement n'est pas normative : la stratégie de recherche de ces vecteurs est ouverte et reste ouverte au choix du concepteur. De ce fait, de nombreuses recherches ont été menées pour optimiser cette stratégie. De plus, dans les standards les plus récents, tels que H.264/AVC, différentes options ont été incorporées. L'étude bibliographique a permis de mettre en évidence d'une part la diversité des codages possibles et d'autre part l'impact de l'algorithme de recherche choisi et des raffinements sélectionnés sur les performances de codage (rapport signal sur bruit). Cette thèse s'inscrit donc dans le contexte général de la compression vidéo et plus précisément dans la partie d'estimation du mouvement.

Objectifs et motivations

Au vue de l'hétérogénéité des standards de codages, de la diversité des média de communication (en particulier des bande−passantes associées) et des terminaux utilisés, il paraît primordial de pouvoir adapter les performances de codage en fonction des évolutions des contraintes liées à l'environnement, voire du contenu de l'image ou des événements détectés dans la scène.

D'autre part, les outils existants combinant rapidité, faible encombrement et faible consommation et qui réalisent la compression sont la plupart du temps basés sur des architectures spécifiques. En effet, devant la demande croissante en puissance de calcul des applications de traitement du signal et particulièrement, des applications de codage d'images fixes et vidéos, il est de plus en plus courant de faire appel à la puissance des algorithmes afin d'accélérer le temps de traitement et respecter les contraintes temps réel.

Notre premier objectif est donc de proposer des algorithmes capables de concurrencer les algorithmes standards existant dans la littérature scientifique.

L'étude exhaustive de toutes les positions du bloc dans la zone de recherche représente la tech-

nique de référence. Il s'agit aussi de la méthode la plus coûteuse en temps de calcul. D'autres stratégies, sub-optimales, utilisent un parcours particulier de la fenêtre de recherche pour limiter le nombre de tentatives de mises en correspondances, et sont de ce fait beaucoup plus rapide.

Notre principale contribution consiste à optimiser l'estimation du mouvement réalisée pour la prédiction de l'image courante par rapport à une image précédemment encodée/décodée selon le choix d'utilisateur ainsi que le type d'application. Il nous a donc été nécessaire d'identifier les éléments pouvant être utilisés au sein d'un modèle commun.

Contributions

Les travaux effectués durant cette thèse s'articulent autour de deux parties de recherche ayant pour but de proposer de nouveaux algorithmes efficaces d'estimation de mouvement flexibles et améliorant les performances des algorithmes existants les plus utilisés[2],[3],[4],[5],[6]. La première partie de nos travaux est orientée vers l'étude des différentes stratégies de recherche afin de proposer une solution d'améliorer l'estimation de mouvement. Dans cette partie, nous identifions principalement les contributions qui apportent des gains significatifs par rapport aux travaux antérieurs :

- Une étude statistique sur un corpus de 12 séquences d'images. Cette étude a mis en relief certain caractéristique du signal vidéo que nous avons exploité.
- La première approche que nous avons nommé par Étoile Diamant effectué d'abord une recherche grossière (Schéma Étoile), puis une autre plus fine (Schéma Petit Diamant).
- Notre deuxième contribution a consisté à réduire la complexité de la recherche du vecteur mouvement à l'aide d'un seuil S

La deuxième partie de nos travaux est consiste à proposer une solution permettant d'optimiser la quantité d'information stockée lors de la vidéo surveillance. Dans cette partie nous proposons une méthode efficace de la détection des objets en mouvement en se basant sur l'estimation de mouvement et la morphologie mathématique afin de servir les systèmes de vidéo surveillance.

Organisation de la thèse

Ce mémoire est composé de deux parties. La première, composée des chapitres 1 et 2, traite de la vidéo en vue de son codage avancé et d'une analyse pratique du standard de compression H.264/AVC, en insistant plus particulièrement sur la partie d'estimation de mouvement. La seconde, composée des chapitres 3 et 4, est dédiée aux implantations des méthodes proposées de l'estimation et la détection de mouvement .

En détail, le manuscrit est organisé comme suit :

- Le premier chapitre est consacré à l'état de l'art des normes de compression vidéo. Durant cette présentation, les concepts de vidéo numérique et les traitements associés aux étapes d'encodage et de décodage de la norme H.264/AVC sont introduits.
- Le second chapitre présente les différents algorithmes ou stratégies de recherche concernant l'estimation de mouvement.
- Le troisième chapitre s'intéresse plus particulièrement à l'étude statistique de la distribution des vecteurs de mouvement effectuée sur un corpus d'images réelles. Nous y trouvons la description des deux méthodes que nous avons élaborées nommées par ED et ED_S.
- Le quatrième et dernier chapitre est consacré à l'introduction d'une nouvelle approche de conception à haut niveau basée sur la détection des objets en mouvement afin d'aider l'opérateur et d'optimiser la quantité d'information stockée lors de la surveillance.

Enfin, ce mémoire s'achève par une partie conclusive sur les principaux apports et résultats obtenus et les perspectives de recherche liées au processus du l'estimation de mouvement.

CHAPITRE 1

COMPRESSION ET CODAGE VIDÉO

1.1 Introduction

Dans ce chapitre, dans une première étape, nous présentons les notions élémentaires sur la compression et le codage vidéo pour la bonne compréhension de notre travail. Une définition de l'image numérique et la notion de résolution de celle-ci sont données (section 1.2.1 et 1.2.2). Une brève description de la vidéo et de la compression sont aussi présentées. Le codec est décrit, on trouvera un bref historique des différentes normes. Le reste de ce chapitre est dédié à l'étude de la norme H.264/AVC. Cette norme est très utilisée et comme nous le verront. Dans cette dernière partie nous montrons l'importance des trois modules de traitement (le module de codage Intra, le module de codage Inter et le module de filtre anti-blocs). Ces trois unités sont les plus lourdes du point de vue temps de calculs. Nous présenterons aussi les opérations d'estimation et de compensation de mouvement qui sont au cœur de notre préoccupation.

1.2 Image numérique et compression vidéo

1.2.1 Image numérique

Une image est stockée en mémoire sous forme de collection de points élémentaires appelés pixels. Nous pouvons considérer une image numérique comme une page de nombres organisés en tableau ou en matrice[7]. Chaque nombre représente les caractéristiques du pixel. La position de chaque pixel peut être exprimée par deux coordonnées sur l'axe horizontal X et l'axe vertical Y comme le montre la figure 1.1 ci-dessous.

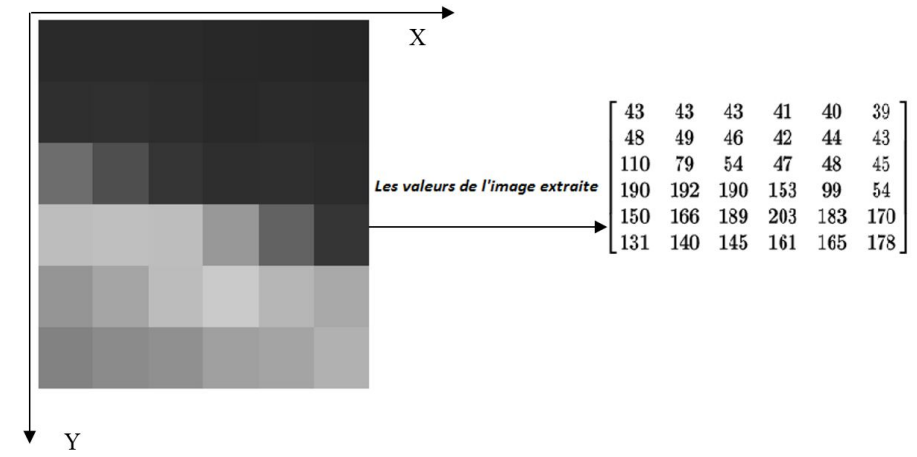


Figure 1.1: Les pixels d'une image.

1.2.2 La résolution d'une image

Une image est composée de pixels (points). Considérons une image de 10 cm sur 10 cm avec une résolution très faible de 10 pixels par cm. Elle est codée sur 100 x 100 = 10000 pixels. Avec une résolution convenable de 100 pixels par cm (un pixel mesure 0,1 mm), elle serait constituée par 1000 x 1000 = 1000000 pixels = 1 M pixels.

En général on utilise l'unité de longueur anglo-saxonne le pouce ou inch. La résolution d'une image s'exprime alors en pixels par pouce (ppp) ou dots per inch (dpi) en anglais. Pour une image avec deux résolutions différentes, la première résolutions ; 18 pixels par pouce soit environ 7 pixels par cm dans ce cas, on observe l'effet de pixellisation (figure 1.2 (a)).

(a) Résolution 1 (b) Résolution 2

Figure 1.2: L'effet de pixellisation.

La deuxième résolutions ; 72 pixels par pouce soit environ 30 pixels par cm (voir la figure 1.2(b)). La dernière résolution correspond approximativement à celle d'un écran d'ordinateur, elle est donc idéale pour visualiser une image sur l'écran[7].

1.2.3 La vidéo

Une image en physique est une représentation d'un objet, produite par la réunion des rayons ou des faisceaux lumineux qui en émanent et se reconstituent sur un miroir, sur un écran ou sur l'œil qui perçoit cette image. Une image numérique est une image échantillonnée selon les deux axes spatiaux. Donc, une image numérique est composée d'une grille de points élémentaires appelés pixels [8]. Le nombre de pixels par image, le nombre de bits pour le codage de chaque pixel et la nature de ces pixels constituent les principales caractéristiques d'une image numérique. Un vidéo numérique est une séquence d'images numériques [9]. L'œil humain a comme caractéristique d'être capable de distinguer environ 20 images par seconde. Ainsi, en affichant plus de 20 images par seconde, il est possible de tromper l'œil et de lui faire croire à une image animée [10]. En plus des caractéristiques d'une image numérique fixe, une vidéo numérique est caractérisée par le nombre d'images par seconde (FPS = Frame Per Second). Le codage vidéo consiste à l'opération d'identification et de représentation d'une vidéo à l'aide d'un code informatique.

1.2.3.1 La vidéo analogique

La caméra balaye l'image bidimensionnelle qu'elle a devant elle par un faisceau d'électrons qui se déplace très rapidement de gauche à droite et plus lentement de haut en bas et produit une tension en fonction du temps. Elle enregistre ainsi l'intensité lumineuse, et à la fin du balayage, on a alors une trame (figure 1.3). Le faisceau revient à l'origine pour recommencer, le récepteur va recevoir cette intensité en fonction du temps, et pour reconstruire l'image, va répéter le processus de balayage [11].

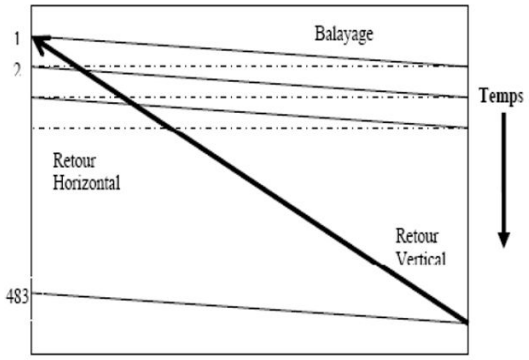


Figure 1.3: Balayage utilisé pour la vidéo et la télévision.

1.2.3.2 La vidéo numérique

La vidéo numérique est une succession d'images formée d'une matrice de pixels. Pour obtenir des images en couleur, il faut utiliser au moins 8 bits ce qui correspond à 256 niveaux de couleur. 8 bits sont utilisés pour chaque couleur Rouge ,Vert et Bleu. Pour la vidéo numérique couleur, soit donc 24 bits par pixel, ce qui convient à environ 16,8 millions de couleurs. Le principe de balayage utilisé est similaire à celui de la vidéo analogique [11].

1.2.4 Compression vidéo

Nous allons maintenant donner un aperçu général du codage vidéo qui permet d'exploiter ces redondances. Nous décrirons pour cela d'abord les différents types de redondance, la manière de décomposer la vidéo en groupe d'images, nous rappellerons un des critères d'évaluations utilisés pour comparer ces techniques, puis nous donnerons un bref rappel historique de

normes de compression et nous finissons par le schéma général d'un codeur vidéo. Le système de compression utilise deux modules complémentaires : le module de compression dit codeur qui consiste à convertir la source de données avant transmission ou stockage en une forme compressée occupant un nombre de bits moindre, et un décompresseur (décodeur) qui convertit la forme compressée en une représentation de la source originale. La paire codeur/décodeur est appelée codec. La compression vidéo consiste à réduire la quantité de données nécessaires pour la représenter en exploitant les redondances d'informations, tout en cherchant que cela soit invisible par le système de vision humain. Cela produit à un compromis entre le taux de compression et la qualité de l'image compressée. L'image devient plus médiocre avec l'augmentation du taux de compression [12].

1.2.4.1 Contraintes

Une vidéo représente une quantité de données importante, et malgré l'augmentation de la puissance des processeurs et des capacités des périphériques de stockage, nous avons besoin de la représenter dans un format plus concis. La qualité d'une vidéo compressée varie très grandement en fonction de la fréquence d'entrée, de la résolution d'entrée, du débit imposé en sortie et de la qualité exigée. La compression des données vidéo numériques sans dégradation significative de la qualité est possible lorsque les séquences vidéo affichent un degré élevé de :

- Redondance spatiale : corrélation entre les pixels voisins.
- Redondance temporelle : corrélation entre les images vidéo.
- Redondance psycho-visuelle : propriétés de la vue humaine [13].

1.2.4.2 La compression de données

1.2.4.2.1 La redondance spatiale Lorsqu'on observe une seule image d'une vidéo, on peut constater qu'elle constitue des zones homogènes plus ou moins grandes dans lesquelles les pixels ont des valeurs très proches, voire identiques : c'est la redondance spatiale (figure1.4). En l'éliminant par codage, il est possible de réduire la quantité d'information à transmettre. La compression des images fixes (norme JPEG Joint Photographic Experts Groupe), utilise ce type de redondance.

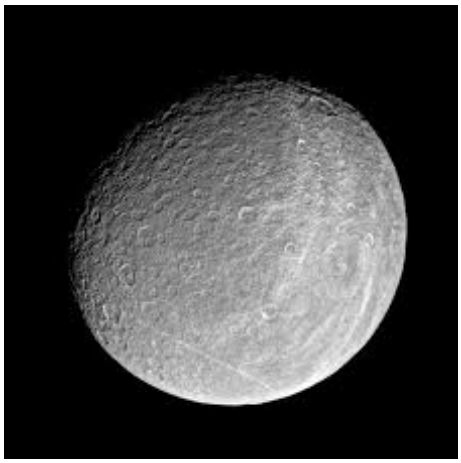


Figure 1.4: La redondance spatiale : la redondance de la couleur noire est très prononcée.

1.2.4.2.2 La redondance temporelle Dans une séquence vidéo, la différence entre une image et la suivante est relativement faible, sauf lors d'un changement de plan. Autrement dit, la position d'un bloc de pixels varie généralement peu d'une image à l'autre.

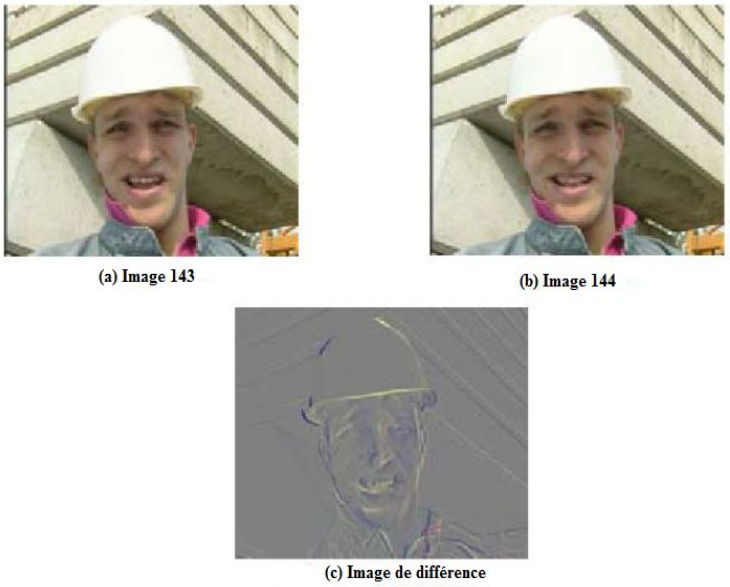


Figure 1.5: La redondance temporelle : la scène avec un fond fixe seul le visage de la persone qui bouge.

La seule différence se trouve dans les parties qui ont subi un mouvement(voir figure 1.5). Ceci peut être démontré par la figure1.5(c) qui représente la différence entre les deux images successives (figure 1.5(a) et figure 1.5 (b)). La partie grise de cette image correspond à la

similitude des deux images (a) et (b). La compression consiste à éliminer les redondances qu'elle soit spatiale ou temporelle.

1.2.4.2.3 La redondance psycho visuelle La redondance psycho-visuelle est généralement réduite. La mesure la plus significative est une résolution réduite du détail des couleurs par rapport au détail de la luminance qui permet un meilleur rapprochement avec les caractéristiques de la perception humaine[13]. Les images vidéo comprennent trois matrices rectangulaires de données de pixels, lesquelles représentent le signal de luminance (luma Y) et deux signaux de chrominance (chroma Cb et Cr aussi appelé U et V) comme le montre la figure 1.6.

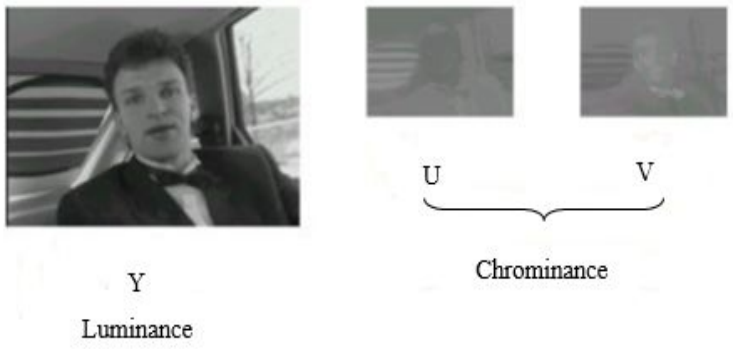


Figure 1.6: Format YUV 4 :1 :1 [13].

Ces matrices correspondent à une représentation décomposée des trois couleurs primaires (RVB) associées à chaque élément d'image [14]. Pour la représentation des pixels vidéo numérique, il existe plusieurs formats comme le montre la figure 1.7.

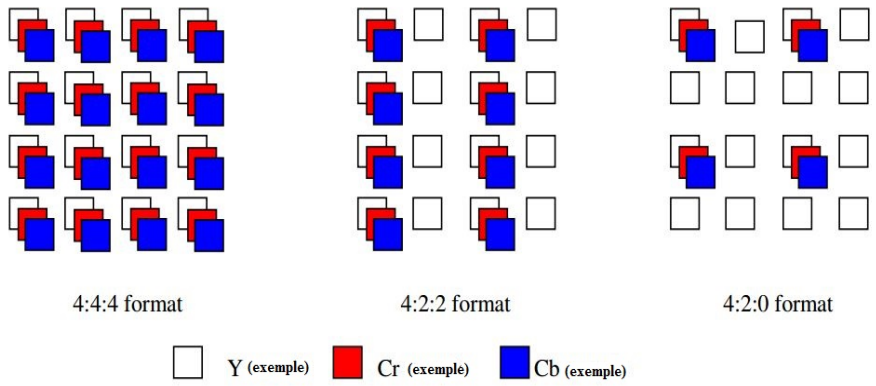


Figure 1.7: Format YUV [14].

1.2.4.3 Décomposition de la séquence vidéo compressée

Une séquence vidéo compressée est généralement structurée en groupe d'images GOP(Groupe Of Pictures). Un GOP est une suite d'images codées suivant trois modes de codage (voir figure 1.8) :

- Les images **Intra**, dites de type **I** : sont des images codées intégralement, sans aucune référence aux images voisines de la séquence vidéo. Les images **I** sont les plus coûteuses en débit mais servent de point de référence dans une séquence vidéo. Chaque changement de plan dans une séquence vidéo commence obligatoirement par une image de type **I**.
- Les images **Inter** prédites, dites de type **P** : Les images « Prédictives » exploitent à la fois la redondance spatiale et la redondance temporelle des images d'une séquence vidéo. Elles sont codées à partir de l'image «**I**» ou «**P**» précédente à l'aide de vecteurs de mouvement obtenu par estimation du mouvement.
- Les images **Inter** prédites bidirectionnelles, dites de type **B** : Elles sont codées avec une estimation du mouvement par rapport à une image précédente et une image suivante. Elles ne servent jamais de référence. Les images de type **I** et **P** servent de référence aux images **P** et **B**, les images de type **B** ne servent jamais de référence [13].

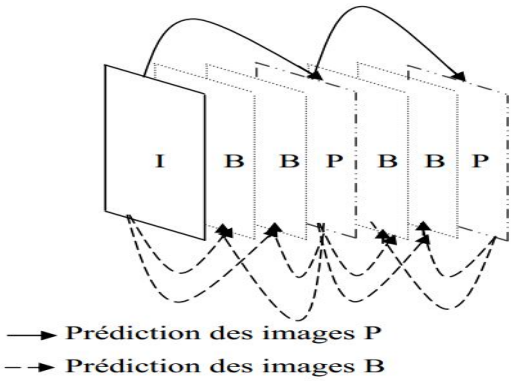


Figure 1.8: Structure d'un groupe d'image (GOP) [13].

1.2.5 Les méthodes de compression vidéo

Le principe fondamental de la compression vidéo est de réduire autant que possible les redondances d'informations dans les données, sans bien sûr que cela ne soit visible de manière flagrante pour l'œil humain. Toute la difficulté est là, dans le dosage entre un taux de compression qui s'améliore en même temps que la qualité d'image devient médiocre.

1.2.5.1 Compression avec perte

Dans le cas de la compression avec perte, les données sont différentes à la sortie du décodeur par rapport à l'entrée du codeur. C'est ce type de compression qui est utilisé en vidéo, car on peut accepter des pertes d'informations qui ne sont pas toujours visibles à l'œil et qui se traduisent par de nets gains de compression. En compression avec pertes, on peut atteindre des taux de compression allant jusqu'à 300 :1 (c'est-à-dire qu'un vidéo de taille 300 Mo, peut attend 1 Mo après la compression) [15].

1.2.5.2 Compression sans perte

Dans le cas de la compression sans perte, les données décodées à l'arrivée par le récepteur sont strictement identiques aux données codées au départ par l'émetteur. Ce type de compression, permettant au mieux un taux de compression de 2 :1, est évidemment insuffisant pour la compression vidéo et ne sera pas l'objet d'étude de cette thèse.

1.3 Codec video

1.3.1 Principe

Un codec vidéo code une source en une forme compressée et la décode pour produire une copie ou une approximation de la source. La séquence originale est représentée par une modélisation, c'est-à-dire une représentation codée le plus efficacement possible et utilisable pour la reconstruction(figure 1.9). Idéalement, la modélisation représente la séquence avec aussi peu de bits que possible et une qualité aussi bonne que possible. Ces deux buts à atteindre sont

évidemment conflictuels. En effet, un débit faible entraîne une image de qualité réduite. Tous les codecs fonctionnent selon le même principe [16].

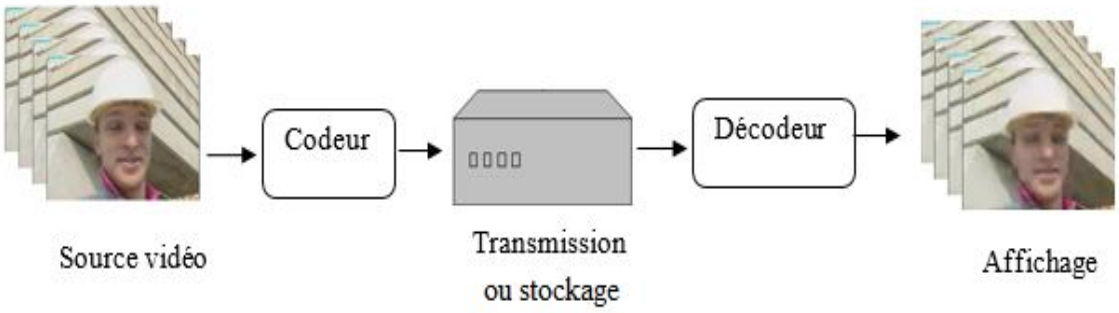


Figure 1.9: Codeur/décodeur.

1.3.2 Standards de compression vidéo

Deux grandes normes sont dans le marché du codage vidéo : la norme H26x et la norme MPEGx (Moving Picture Experts Group). La première développée par ITU−T(International Telecommunication Union−Telecommunication) et la deuxième par ISO/CEI(International Organization for Standardization/International Electrotechnical Commission) pour le traitement et le codage audio/vidéo, notamment les familles MPEG et H.26x. Ce sont les objectifs du codage qui déterminent la structure et les principes de l'encodeur. La norme H26x est destinée initialement à la vidéo conférence qui est caractérisée par des mouvements lents et des arrières plans statiques, tandis que la norme MPEGx est destinée aux applications de la vidéo numérique et de l'informatique ainsi qu'aux applications de la télévision numérique.
Ci-après un extrait historique de ces normes :

- **MPEG 1 :** La norme MPEG1, enregistrée à l'ISO sous le code ISO/IEC 11172, a été finalisée par le groupe MPEG en 1992. C'est leur premier standard audio et vidéo utilisé plus tard comme standard des Vidéo CDs [17]. Il inclut le populaire format audio MPEG−1 Layer 3 (MP3).

- **MPEG 2 :** La norme MPEG−2 est enregistrée à l'ISO sous le code ISO/IEC 13818, et a été finalisée en 1996. Elle a été développée pour la compression de la vidéo de qualité

télévision à un débit de 4 à 6 Mbits/s (ce qui correspond à un canal de télévision normal). Un peu plus tard, MPEG−2 intégra la compression de la télévision à haute définition (HDTV). Aujourd'hui, MPEG−2 est aussi le format utilisé pour stocker les films sur DVD [18].

- **MPEG 3 :** a été conçu pour la prise en charge des signaux (HDTV) (La télévision à haute définition) à des débits de 20 à 40 Mbits/s. Il a vite été observé que des résultats semblables pouvaient être obtenus par de légères modifications de MPEG−2. Perdant de son intérêt, MPEG-3 a été abandonné.
- **MPEG 4 :** C'est en 1994 que MPEG (Moving Picture Experts Group) a développé MPEG−4. Le protocole est devenu un standard officiel ISO/IEC en janvier 1999. Il vise à fusionner trois domaines : l'informatique, la télécommunication et la télévision. La norme MPEG-4 est radicalement différente des normes précédentes. Elle initialise une vidéo orientée objets : une séquence est une scène hiérarchisée en objets média, chaque objet est composé d'un ensemble de paramètres (son, forme, position,...) [19].
- **MPEG 7 :** introduite en 1997, est une norme de représentation du contenu des documents multimédia. La norme est une description standardisée des données permettant une interprétation informatique. Les outils de description du contenu visuel sont groupés en divers catégories : couleur, texture, forme, contour, mouvement [20].
- **H.261 :** Cette norme a été développée par l'ITU en 1990 pour l'appliquer en vidéo conférence et en visiophonie sur les réseaux RNIS (Réseau Numérique à Intégration de Service) en anglais ISDN (Integrated Services for Digital Network). Elle offre une transmission des images animées sur des supports à multiplexage de 64 Kbit/s. Elle utilise des séquences de 30 images par seconde de type I et P. Les images I sont codées selon le principe JPEG (Joint Photographic Experts Groupe). Les images P sont codées selon le principe de compensation de mouvement [21].
- **H.263 :** C'est un standard de l'ITU (International Telecom Union), développé en février 1995. Il a été dédié aux communications à faible débit (moins de 64 Kbit/s). Mais le standard a été ensuite développé pour être utilisé dans des applications à débit élevé et

a pu remplacer le H261 dans plusieurs applications surtout dans la vidéo conférence et a dominé le domaine de la vidéo sur IP (Internet Protocol) [22].

– **H.264/MPEG−4 AVC** L'arrivée du codage H.264/AVC (Advanced Video Coding) a permis d'atteindre une meilleure efficacité des étapes de compression à qualité visuelle équivalente. Cette augmentation des performances de compression est en partie due à une plus grande souplesse dans le partitionnement de l'image en blocs élémentaires vis à vis des normes antérieures. En effet, à partir des avancées proposées dans cette norme, il est possible de regrouper des blocs élémentaires en macro blocs de taille et de forme variables (bandes, rectangles, carrés de résolution et de dimension variables). De plus, la taille des blocs élémentaires sur lesquels sont estimés les mouvements entre les images successives a été réduite de 8x8 pixels à 4x4 pixels. Il résulte de ces deux évolutions une meilleure adéquation entre l'image compressée et les données d'origine. De plus, en ce qui concerne le codage des redondances temporelles, la norme H.264/AVC permet de rechercher un bloc similaire se trouvant à une distance temporelle beaucoup plus grande (jusqu'à une distance de 16 images). Actuellement, cette norme reste couramment utilisée par les producteurs et les diffuseurs de contenus vidéo. Toutefois ces principales améliorations possèdent un contre coût important sur la complexité des processus d'encodage et de décodage des flux vidéo [23].

1.4 Étude de la norme H.264/AVC

La chaîne de codage vidéo H.264/AVC (Advanced Video Coding) illustrée par la figure 1.10 ci-dessus est semblable à celle des autres normes telles que MPEG−2 et MPEG−4. Il s'agit d'un hybride de prédiction temporelle (mode Inter) et de prédiction spatiale (mode Intra). En effet, la première image d'une séquence est forcément Intra codée. Pour les autres images d'une séquence on peut utiliser soit un codage inter, soit un codage intra. Le standard H.264/AVC permet d'obtenir des meilleures performances de compression par rapport à ses prédécesseurs grâce à de nombreuses spécificités de compensation de mouvement (codage inter−image) pour utiliser ces modes de prédiction, l'estimation de mouvement doit prendre en compte leur impact sur les algorithmes ainsi que sur la complexité de calcul [24]. Nous allons détaillés les unités

principales de codeur vidéo déjà mentionnées.

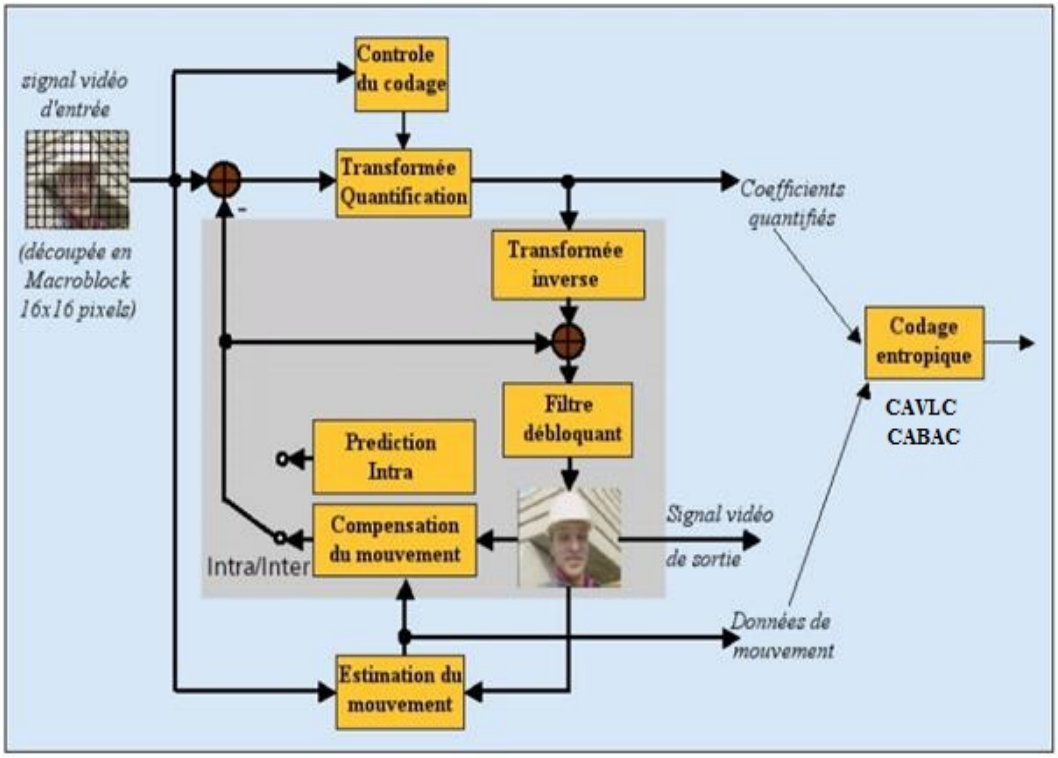
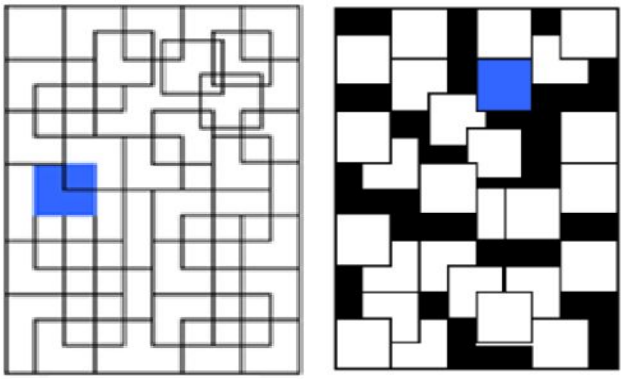


Figure 1.10: Schéma de codage principal du standard H.264/AVC [24].

1.4.1 Prédiction inter-image

1.4.1.1 Prédictions avant et arrière (*forward* and *backward*)

La prédiction de mouvement permet d'estimer le mouvement des blocs entre deux images consécutives. Le mouvement calculé va permettre de prédire l'image f_{k+1} à partir des pixels de l'image f_k et les vecteurs mouvements, deux types de prédiction sont possibles :
La prédiction *avant* (forward) et la prédiction *arrière* (*backward*).
La prédiction *forward* consiste à trouver les blocs de l'image f_k dans l'image f_{k+1}(voir figure1.11 (b)). La mise en correspondance des blocs avec prédiction forward répond à la question : où va le bloc courant ?
En revanche, la prédiction backward consiste à trouver les blocs de l'image f_{k+1} dans l'image f_k(voir figure 1.11 (a)). Dans ce cas la question est : d'où vient le bloc courant ?



(a) (b)

Figure 1.11: Résultats de Block-matching avec prédiction backward et forward : prédiction backward de l'image f_k en (a) et prédiction forward de l'image f_{k+1} en (b) [25].

Le résultat issu des deux approches est différent. En effet, l'image prédite par prédiction forward présente des trous puisque les blocs n'ont pas un mouvement unique pour toute l'image. Lors de la prédiction, certains blocs se recouvrent et laissent apparaître des zones non prédites et par conséquent, noires (figure 1.11 (b)). Ce problème est entièrement résolu par l'utilisation de la prédiction *backward* (figure 1.11(a)) qui ne laisse entrevoir aucun trou dans la prédiction [26]. La prédiction Inter (type P) est utilisée pour réduire la corrélation temporelle avec l'aide de l'estimation et la compensation de mouvement. Dans H.264/AVC, l'image courante peut être subdivisée en macro-blocs ou en blocs plus petits. Pour le macro-bloc (16x16), il peut y avoir jusqu'à 7 modes : 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 et 4x4. Cependant, le choix de la taille des partitions dépend essentiellement des caractéristiques de la séquence d'entrée. En général, les grandes partitions sont choisies pour couvrir les zones homogènes de la même image et les partitions de petites tailles sont utilisées pour des zones avec beaucoup de détails. Le processus de la prédiction inter, peut faire la segmentation de la représentation des mouvements aussi petite que la taille des blocs 4x4, en utilisant des vecteurs de mouvement avec une précision d'un quart de l'échantillon pour la luminance. La compensation en mouvement des sous-pixels peut donner de meilleures performances de compression que la compensation entière de pixel, au dépend d'une complexité accrue [27]. La compensation de mouvement utilise des similitudes qui existent entre l'image courante d'entrée et l'image encodée précédemment. Cette image déjà codée appelée image de référence qui est en fait une image déjà reconstruite.

1.4.2 Prédiction intra-image

Contrairement à la prédiction inter, la prédiction intra-image (type I) n'utilise pas d'images de référence mais uniquement l'image courante. Il s'agit donc d'exploiter la redondance spatiale présente dans toute image. En effet, le principe de base est que dans un certain voisinage, la structure de l'image varie peu globalement. Bien sûr, ceci dépend de la taille de l'échantillon considéré, c'est pourquoi chaque taille de bloc correspond à un traitement spécifique. De fait, pour la luminance, la prédiction peut être réalisée sur des blocs 4x4 ou des macro-blocs 16x16. De plus, bien que la luminance et la chrominance n'aient pas la même résolution, la chrominance n'est codée que par blocs 8x8. Un bloc prédit est donc construit à partir de ces voisins déjà codés et reconstruits. Il est ensuite soustrait au bloc original et le bloc différence est codé et transmis. Toute la procédure de prédiction doit ensuite être insérée au flux codé, ce qui nécessite de l'information supplémentaire dont l'importance ne doit pas être négligée. Cette section présente les différentes procédures de prédiction intra-image, suivant la taille de bloc considérée, et la manière dont la procédure de prédiction est codée dans le flux [28]. Chaque macro-bloc (taille 16x16 pixels) est codé par le codeur H.264/AVC en utilisant deux grandes unités de transformation.

1.4.3 La transformation

L'utilisation de transformations mathématiques permet de remplacer les valeurs produites par la numérisation (l'image résiduelle) par des données de nature différente qui peuvent se révéler plus faciles à compresser. Parmi les transformations mathématiques les plus connues : La Transformée Cosinus Discrète (TCD) qui est basée sur des blocs produit une matrice de 64 coefficients (8x8), rangés selon les fréquences croissantes selon les axes horizontal et vertical. Une des principales différences entre le codec H.264 et les précédents est qu'il n'utilise plus le TCD 8x8, H.264/AVC utilise une transformation beaucoup plus simple et réversible, qui sépare chaque entrée ou d'une erreur macro-bloc dans une série de blocs de 4x4 pixels puis tout simplement de convertir les blocs dans un état plus favorable à la compression. Elle diminue la complexité de traitement, spécialement pour les processus courts. La transformée inverse de

H.264/AVC est définie de manière exacte (précision entière) pour éviter les divergences dues aux arrondis [29].

Cette étape ne compresse donc pas les données. On n'a que des pertes de précision dues aux erreurs d'arrondies lors des calculs. On considère la transformation TCD comme une étape conservatrice qui prépare les données à la phase non conservatrice du processus de la quantification [30].

1.4.4 Quantification

Pour quantifier les coefficients de la transformée, la norme H.264/AVC utilise une quantification scalaire, c'est l'étape où la majorité de la compression est atteinte. C'est également le stade où la majorité des renseignements peuvent être perdus. La quantification n'est que le processus de réduction du nombre de bits nécessaires au stockage d'une valeur entière d'où la diminution de la précision de la qualité. Les changements rapides dans les hautes fréquences peuvent être imperceptibles, c'est pourquoi on peut les éliminer [31]. Ce processus est contrôlé par un paramètre appelé Qp, où Qp peut prendre une valeur comprise entre 0 et 51. Dans les normes vidéo précédentes, le pas de quantification augmente par un pas fixe, cela entraîne des zones inaccessibles pour certains quantificateurs. Si Qp est défini à 0, alors l'unité de quantification exécute peu de traitement sur la transformation de données, ce qui signifie que peu de données sont perdues, la qualité reste élevée mais la compression atteinte est faible. Comme Qp augmente en valeur l'unité de quantification commence à supprimer des informations. Cependant, l'encodeur est conçu pour supprimer uniquement les plus insignifiants détails souvent la perte d'information est imperceptible à l'oeil humain. Comme Qp augmente vers la valeur maximale 51, de plus en plus l'information est rejetée et la qualité doit être sacrifiée. Toutefois, la compression va augmenter de manière significative, comme l'augmentation Qp [32].

- La quantification est donc la phase qui provoque une dégradation dans l'image reconstruite mais c'est aussi cette opération qui permet d'obtenir des taux de compressions beaucoup plus importants.

1.4.5 Filtre débloquant

Un défaut du codage axé sur le bloc est la visibilité de la structure en blocs. Les bords sont en général reconstitués avec moins de précision que les pixels intérieurs : la pixellisation est l'un des artefacts les plus visibles des méthodes de compression actuelles. Pour cette raison, la norme H.264/AVC définit un filtre de «déblocage» adaptatif en boucle où la puissance du filtrage est contrôlée par les valeurs de plusieurs éléments syntaxiques. La pixellisation est alors réduite sans affecter outre mesure la netteté du contenu et la qualité subjective est considérablement améliorée (voir figure 1.12). En même temps, le filtre réduit le débit binaire en général de 5 à 10% tout en produisant la même qualité objective que la vidéo non filtrée [33].

(a) Non filtré. (b) Filtré.

Figure 1.12: Effet de déblocage en H.264/AVC [34].

1.4.6 Codage entropique

L'opération de codage entropique est placée à la fin de la chaîne de codage vidéo. Elle consiste à représenter l'information de manière plus compacte et sans perte en un flux binaire compressé afin de le mettre à disposition du canal de transmission. Son principe est statistique : le codeur va attribuer des codes courts aux symboles les plus courants et des codes plus longs à ceux qui sont moins fréquents. La longueur de ce code dépend de la fréquence d'apparition du symbole considéré. Dans le standard H.264/AVC, les deux principaux codeurs entropiques qui ont été implémentés sont le CABAC (Context Adaptive Binary Arithmetic Coding) et le CAVLC (Context Adaptive Variable Length Coding)[35].

1.5 Estimation et compensation de mouvement

La partie dite d'estimation du mouvement, nous allons maintenant détailler plus avant cette partie de l'encodage. L'estimation de mouvement consiste à trouver dans l'image de référence un macro-bloc correspondant le plus possible semblable au macro-bloc courant. Dans l'image de référence, on définit une zone de recherche dans laquelle le macro-bloc à estimer est comparé à tous les macro-blocs figure 1.13. La différence de position entre les deux macro−blocs (courant et référence) est appelée vecteur du mouvement [1]. Dans une séquence vidéo, la différence entre une image et la suivante est relativement faible, sauf lors d'un changement de plan : c'est la redondance temporelle. Le but du codage inter-image ou compensation de mouvement est de prédire l'image à coder en fonction d'une image de référence et d'un champ de vecteurs de mouvement. Ensuite, le résidu est encodé avec les techniques présentées ci-dessus par la transformation et la quantification. De manière générale, le volume des données à transmettre (champ de vecteurs et résidus) est réduit par rapport à un codage intra−image, c'est−à−dire sans exploitation de la redondance temporelle.

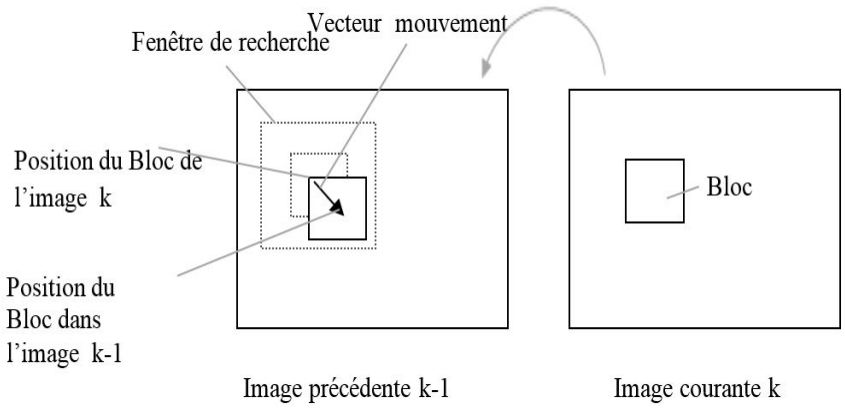


Figure 1.13: Comparaison entre les macro-blocs dans l'image courante et référence (Backward).

La compression par compensation de mouvement consiste, pour calculer une image prédictive (P) (reconstruit), à ne coder que les vecteurs de mouvement résultants de l'estimation de mouvement et de l'image de référence. L'image prédite soustraite l'image courante pour donner les erreurs résiduelles(voir la figure 1.14). Ces erreurs sont ensuite transformées, quantifiées

et codées avant d'être transmises. Donc, le but de compensation du mouvement est de fournir une information supplémentaire pour ajouter l'information résiduelle nécessaire à la bonne prédiction de l'image. Plus la prédiction est performante, plus l'erreur résiduelle est petite [36].

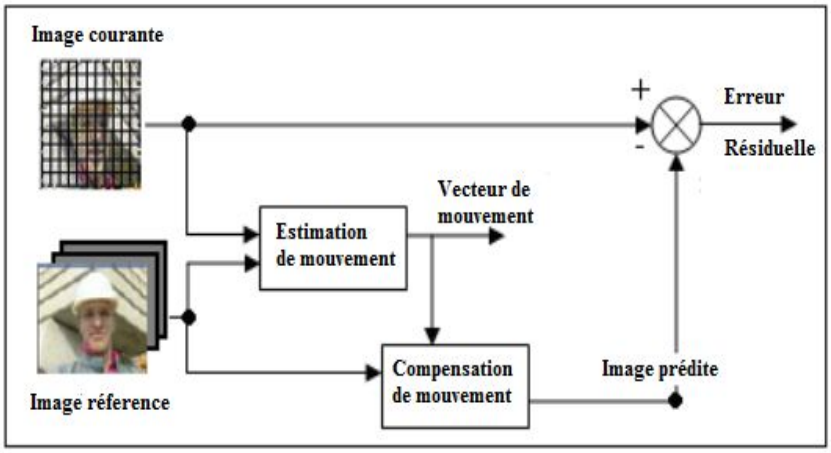


Figure 1.14: Estimation et compensation de mouvement.

En outre, la compensation de mouvement peut être effectuée à partir d'une ou plusieurs images de référence afin d'améliorer la prédiction. Chaque bloc peut être reconstruit à partir d'une seule image de référence ou de deux références. On parle alors de biprédiction. La compensation de mouvement permet de réduire considérablement la bande passante nécessaire à la transmission d'une séquence vidéo.

1.5.0.1 Taille des blocs

La norme H.264 réduit le débit des vidéos en ajoutant des modes de codage par rapport aux anciens standards. La compensation de mouvement peut être faite sur des blocs de taille variable. Les macro-blocs 16×16 peuvent être divisés en 16×8, 8×16 ou 8×8 et les sous-partitions 8×8 peut être à nouveau divisées en 8×4, 4×8 ou 4×4 (voir figure1.15). Le choix de petits blocs améliore la précision de la compensation de mouvement mais nécessite la transmission d'un plus grand nombre de vecteurs. Un algorithme de décision efficace doit donc être mis en place. Il peut faire partie intégrante de l'estimateur ou bien être séparément exécutée. Dans ce dernier cas l'estimateur de mouvement fourni les vecteurs pour toutes les tailles de blocs [35].

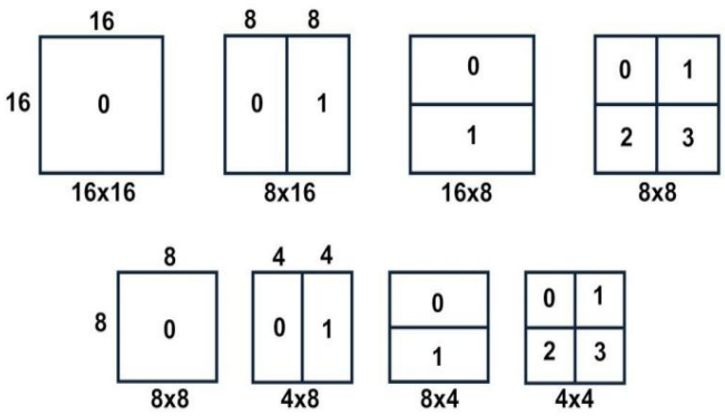


Figure 1.15: Décomposition d'un macro-block.

1.5.1 Critères d'évaluation

Nous utilisons pour comparer les différents paramètres plusieurs critères d'évaluation couramment utilisés dans l'estimation du mouvement. Ces critères peuvent être séparés en deux catégories : les critères qualitatifs et les critères de complexité.

1.5.1.1 Critères de mesure de la qualité

1.5.1.1.1 Peak Signal to Noise Ratio ($PSNR$) Le critère que nous avons utilisé est le PSNR défini par l'équation suivante :

$$PSNR = 10\log_{10}\left\{\frac{PS^2}{MSE}\right\}(dB) \tag{1.1}$$

Où PS est la valeur maximale de pixels et MSE est l'erreur quadratique moyenne définie par :

$$MSE = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(f_k(i,j) - \hat{f}_k(i,j)\right)^2}{N \times M} \tag{1.2}$$

Avec :

$f_k(i,j)$ l'image courante, $\hat{f}_k(i,j)$ l'image reconstruite et N, M respectivement le nombre de lignes et de colonnes de l'image. La qualité de l'image reconstruite augmente avec le $PSNR$ et inver-

sement. Il existe d'autres critères d'évaluation de la qualité d'image reconstruite basé sur les caractéristiques de la vision humain tels que :$PSNR\_HVSM$[37],$SSIM$[38].

1.5.1.1.2 PSNR_HVSM PSNR_HVSM est un critère qui avait été conçu pour améliorer les performances de PSNR et MSE. Il est basé sur l'information de contenu de l'image et les caractéristiques de système de la vision humain pour évaluer la qualité d'image. Cependant, le PSNR et MSE ne peuvent pas obtenir de bons résultats par rapport aux qualité subjective, surtout pour les images corrompues avec bruit haute fréquence, bruit impulsionnel, bruit additif gaussien, flou gaussien, etc.

$$\delta\left(i,j\right)=\left[f(i,j)-\hat{f}(i,j)\right] \tag{1.3}$$

La PSNR_HVSM consiste à diviser l'image en blocs de 8 x 8 pixels non-cumul. Ensuite, la $\delta\left(i,j\right)$ (la différence) entre le bloc original et le bloc reconstruit est calculé avec les coefficients de DCT. ensuite les coefficients DCT est encore multipliée par un contraste de masquages métriques (CM). Le résultat est ensuite pondéré par les coefficients de la fonction de sensibilité de contraste CSF_{Coef} comme suit, donc l'équation (1.3) peut être réécrite comme suit,

$$\delta_{PSNR\_HVSM}\left(i,j\right)=\left(\delta\left(i,j\right)\cdot CM(i,j)\right)\cdot CSF_{Coef}\left(i,j\right) \tag{1.4}$$

Par conséquent, une nouvelle mesure MSE pour PSNR_HVSM peut être définie comme suit,

$$MSE_{PSNR\_HVSM}=\frac{1}{M\times N}\sum_{I=1}^{M/8}\sum_{J=1}^{N/8}\left(\sum_{i=1}^{8}\sum_{j=1}^{8}\left(\delta_{PSNR\_HVSM}\left(i,j\right)\right)^{2}\right) \tag{1.5}$$

Où (I,J) est la position du bloc 8 x 8 à l'image, (i, j) est l'emplacement d'un pixel dans le bloc de 8 x 8. Et PSNR_HVSM peut être calculé en remplaçant le MSE_{PSNR_HVSM} dans l'équation (1.1).

1.5.1.1.3 SSIM SSIM permet d'évaluer la qualité d'image reconstruite, ce critère est calculé pour chaque pixel de l'image et est compris entre 0 et 1. Cette valeur donne un niveau de similarité des pixels comparés. Plus elle est proche de 0, plus les pixels sont différents et donc, plus le pixel est déformé. Il s'agit de la luminosité, du contraste, et de la structure du signal.

Ces valeurs sont ensuite combinées pour donner l'indice SSIM [38]. Notons que ces trois critères sont indépendants, un changement au niveau de la luminosité ou du contraste n'affectera pas la structure de l'image. Nous allons maintenant définir les trois fonctions permettant la comparaison des trois critères mentionnés précédemment pour ensuite donner la définition de l'indice SSIM.

La comparaison des luminosités est réalisée avec la formule suivante :

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{1.6}$$

Où x et y sont l'intensité moyenne de chacun des signaux, ce qui donne une estimation de la luminosité, et $C1$, une constante de stabilité.

Intensité moyenne du signal x :

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1.7}$$

$$C_1 = (K1 * L)^2 \tag{1.8}$$

$$K1 \ll 1 \tag{1.9}$$

L est le rang dynamique des valeurs des pixels (255 pour une image de niveau de gris codé sur 8 bits).

La fonction de comparaison de contraste prend la forme suivante :

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{1.10}$$

Où σ_x et σ_y sont les écart-types de chacun des signaux, ce qui donne une estimation du contraste, et $C2$, une constante de stabilité.

Écart type (la racine carrée de variance) du signal original x :

$$\sigma_x = (\mu_x(X^2) - \mu_X^2)^{1/2} \tag{1.11}$$

$$C_2 = (K2 * L)^2 \tag{1.12}$$

$$K2 \ll 1 \tag{1.13}$$

La fonction de comparaison de structure est défini comme suit :

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} = \frac{\mathrm{cov}(x,y) + C_3}{\sigma_x \sigma_y + C_3} \tag{1.14}$$

Où $C3$ est une constante de stabilité. Donc la combinaisons de ces trois comparaisons détermine l'indice de similarité structurelle (SSIM) entre deux signaux x et y (on prend $C3 = C2/2$ pour simplifier l'expression) : Avec :

Covariance du couple (x, y).

$$\mathrm{cov}(x,y) = \mu_{xy} - \mu_x.\mu_y \tag{1.15}$$

Intensité moyenne du produit $x. * y$

$$\mu_{xy} = \sum_{i=1}^{N} x_i.y_i \tag{1.16}$$

L'indice SSIM mesure la différence entre les signaux x et y, est défini de la manière suivant :

$$SSIM\,(x,y) = l\,(x,y)\,.c\,(x,y)\,.s\,(x,y) \tag{1.17}$$

Il combine les fonctions de comparaison de luminosité, de contraste, et de structure en faisant le produit de ces trois dernières.

1.5.1.2 Critère de mesure de la complexité

1.5.1.2.1 Nombre de Points Calculés (NPC) Pour la recherche du meilleur bloc, l'algorithme Full Search calcule 15x15=225 SAD (Sum of Absolute Differences) et donc 225 points sont auscultés [39]. La plupart des algorithmes d'estimation de la littérature scientifique que nous avons étudiés réduisent la complexité de Full Search, ils consistent à trouver pour chaque bloc de l'image à prédire le bloc minimisant le critère de comparaison (SAD) tout en tentant d'en tester le moins possible [40]-[41]. Nous avons défini pour la première fois la notion de

Nombre de Points Calculés (NPC) pour mesurer la complexité des algorithmes. NPC d'un bloc dans une fenêtre de recherche n'étant que le nombre de SAD (**S**um of **A**bsolute **D**ifferences) calculés. Pour l'image le NPC moyen est défini de la manière suivante :

$$\mathrm{NPC}_I = \frac{SAD_n}{nb} \tag{1.18}$$

Avec :

SAD_n :Le nombre total de SAD calculés entre une paire d'images.

nb :Le nombre de blocs obtenus en subdivisant l'image en bloc de taille identique TxT.

Et pour une séquence d'images nous avons défini le Nombre de Points Calculés moyen (NPC_{moy}) comme la moyenne de tout les NPC_I de chaque paire d'images de la séquence.

Une bonne manière de se rendre compte de l'efficacité de l'algorithme est d'étudier le nombre moyen de points calculés NPC. Donc la complexité de l'algorithme est d'autant plus réduite que NPC_{moy} est faible.

1.6 Conclusion

Nous avons présenté dans ce premier chapitre, les principales notions de base de l'image et la vidéo numérique. Ce chapitre dans sa premier partie, rappelle les principales généralités de la vidéo numérique et l'historique des différents standards ainsi que les critères d'évaluations. Dans la deuxième partie nous avons détaillé la norme H.264/AVC ainsi que le fonctionnement de principales unités. La dernière partie de ce chapitre est dédiée à la phase de l'estimation et la compensation de mouvement de la norme H.264/AVC d'une manière générale, et qui seront plus étudiés en détail dans le chapitre suivant, en présentons les algorithmes d'estimation de mouvement les plus utilisés.

CHAPITRE 2

ESTIMATION DU MOUVEMENT : ÉTAT DE L'ART

2.1 Introduction

Comme nous l'avons vu au cours du chapitre précédent, l'opération d'estimation de mouvement est cruciale. En effet, d'une part la qualité de l'estimation influe sensiblement sur les performances de la compression et d'autre part c'est l'opération qui nécessite le plus de puissance de calcul. Dans ce chapitre nous décrivons brièvement l'ensemble de méthodes d'estimation de mouvement que nous classons en trois catégories : les méthodes différentielles, les méthodes fréquentielles et les méthodes de mise en correspondance de blocs "Block Matching Algorithm"(BMA). Parmi ces techniques, nous nous sommes intéressés aux méthodes les plus importantes de type BMA qui sont actuellement adoptées dans plusieurs normes de compression. Nous trouvons dans la suite de ce chapitre l'étude et la description des méthodes Four Step Search (4SS), la Diamond Search (DS), la Hexagon Search (HEX), la Cross Diamond Search (CDS) et enfin la Novel Octagon Cross Diamond Search (NOCDS). Nous proposons, pour l'étude comparative de ces méthodes, le paramètre NPC que nous pensons adéquat pour estimer la complexité de chacun des algorithmes que nous avons étudiés.

2.2 Techniques d'estimation de mouvement

Les techniques d'estimation de mouvement consistent a mesurer le flot optique représenté dans les variations de la luminance entre deux images[25]. La plupart des techniques d'estimation du mouvement reposent sur les principes suivants :

- La luminosité est uniforme le long des déplacements. Les variations dans le temps de la luminosité sont négligeables. Elles correspondent à un changement visuel mais pas à un réel déplacement.
- Les problèmes d'occlusion sont négligés. En effet, ces problèmes se produisent lorsqu'une partie de l'arrière-plan cachée apparaît. Aucune zone de l'image de référence ne lui correspond normalement, ce qui est rarement le cas dans une séquence vidéo réelle.

La plupart des méthodes d'estimation du flot optique reposent sur une hypothèse fondamentale : l'intensité lumineuse (I) se conserve entre deux images successives à l'instant t(temps). Cela s'écrit sous la forme générale :

$$I\left((x+dx, y+dy), t+1\right) - I\left((x,y),t\right) = 0 \tag{2.1}$$

Où (x, y) sont les positions des pixels et (dx, dy) représentent les déplacements recherchés. Il est également possible de formuler le problème sous forme différentielle, ce qui conduit à :

$$\frac{dI\left(x\left(t\right), y\left(t\right), t\right)}{dt} = \frac{\partial I}{\partial x}.\frac{dx}{dt} + \frac{\partial I}{\partial y}.\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{2.2}$$

Où dx, dy représentent les déplacements recherchés.

Soit l'équation du flot optique :

$$\left(\nabla \mathrm{I}\right)^{\mathrm{T}} \omega + I_t = 0 \tag{2.3}$$

Avec $\omega = [ux, uy]^T$ est la vitesse recherchée au position des pixels (x, y) et $\nabla I = [I_x, I_y]^T$ le gradient spatial de l'intensité lumineuse, I_t la dérivée temporelle. Cette contrainte est insuffisante pour déterminer le flot complet ω car le problème est mal posé : on dispose d'une équation linéaire pour deux inconnues. Il faut donc émettre une hypothèse supplémentaire. Les

méthodes détaillées dans la suite de cette partie proposent différentes solutions au problème.

2.2.1 Méthodes différentielles

Les méthodes basées sur l'approche différentielle consistent à résoudre un problème d'estimation (local ou global) en minimisant une fonctionnelle, généralement basée sur l'équation (2.3) à laquelle on ajoute une contrainte pour particulariser les solutions.

2.2.1.1 Méthodes globales

Ce type d'approche consiste à minimiser sur le domaine entier de l'image une fonctionnelle prenant en compte l'équation (2.3) du flot optique ainsi qu'un terme de lissage, c'est-à-dire en ajoutant une contrainte de régularisation portant sur le gradient, le laplacien (ou ordre supérieur) du champ de vitesse. La principale méthode globale de calcul du flot optique a été développée en 1980 par Horn et Schunck [42] et consiste à minimiser sur l'ensemble de l'image la fonctionnelle :

$$J_{HS} = \iint (\left((\nabla \mathrm{I})^{\mathrm{T}} \omega + I_t\right)^2 + ((\nabla v_x)^2 + (\nabla v_y)^2))dxdy \tag{2.4}$$

Cette régularisation se justifie par le fait que les vitesses voisines sont presque identiques. Il existe des variantes de cette méthode utilisant d'autres opérateurs de régularisation [43]. Les résultats présentent toutefois tous les mêmes types de défauts [44] : Certes le flot est dense, mais lisse et très bruité. Par ailleurs, le caractère global de l'approche ampute le résultat des petits mouvements de l'image. La conséquence de ceci est un lissage excessif du flot (grand mouvement d'ensemble observé).

2.2.1.2 Méthodes locales

Les méthodes locales consistent à prendre en compte des hypothèses supplémentaires sur un domaine de taille réduite pour particulariser le flot optique. On minimise alors un critère sur un petit domaine, et on obtient ainsi le flot optique de ce petit domaine. La méthode locale la plus célèbre est celle développée par Lucas et Kanade [45] : la vitesse locale est supposée

constante sur un voisinage spatial Ω, on minimise alors sur le domaine la fonctionnelle :

$$J_{LK} = \sum_{\Omega} W^2 [\nabla I . \vec{\omega} + I_t]^2 \tag{2.5}$$

W est une fenêtre locale, pouvant également être interprétée comme la pondération du critère des moindres carrés. On donne généralement une importance plus grande au pixel central (filtrage de type gaussien, facultatif). Les méthodes différentielles locales sont intéressantes car hautement parallélisables (chaque calcul sur une petite fenêtre est indépendant des autres). Les résultats sont par ailleurs moins sensibles au bruit et permettent le calcul de mouvements locaux, notamment à l'aide d'une implémentation pyramidale [46].

2.2.2 Méthodes fréquentielles

Une autre catégorie de méthodes s'appuie sur les manipulations possibles dans le domaine fréquentiel. La transformée de Fourier de l'équation (2.3) du flot optique nous donne :

$$v_x f_x + v_y f_y + f_t = 0 \tag{2.6}$$

Où f_x, f_y sont les fréquences spatiales et f_t la fréquence temporelle. Les Filtres de Gabor sont des filtres 3D formés par le produit d'une gaussienne et d'une fonction trigonométrique, soit :

$$g(x, y, t) = \frac{1}{(2\pi)^3 \sigma_x \sigma_y \sigma_t} e^{\frac{-x^2}{2\sigma_x^2} - \frac{-y^2}{2\sigma_y^2} - \frac{-t^2}{2\sigma_t^2}} . \cos\left(2\pi \left(f_{x0} x + f_{y0} y + f_{t0} t\right)\right) \tag{2.7}$$

Où :$(\sigma_x, \sigma_y, \sigma_t)$ est l'écart type de la gaussienne et (f_{x0}, f_{y0}, f_{t0}) la fréquence centrale du filtre. Ces filtres sont des passe bande. La transformée de Fourier de ce filtre est une gaussienne centrée en (f_{x0}, f_{y0}, f_{t0}) et d'écart type $(\frac{1}{\sigma_x}, \frac{1}{\sigma_y}, \frac{1}{\sigma_t})$

2.2.2.1 Méthodes par filtrage

Plusieurs méthodes proposent une combinaison de filtrages pour évaluer différents ordres de grandeur de vitesse, soit par une approche multi-résolution (pyramide de Heeger), soit en filtrant l'image par plusieurs filtres de Gabor orientés différemment de manière à être en pré-

sence d'un système surdéterminé pouvant être résolu immédiatement (Weber et Malik) [47]. De même, Fleet et Jepson [48] proposent l'utilisation de la phase obtenue après convolution de l'image avec une famille de filtres de Gabor. Il existe également des approches utilisant des familles d'ondelettes particulières [49] (équivalentes à des projections sur des filtres particuliers). L'inconvénient de toutes ces méthodes est l'utilisation excessive du filtrage, qui entraîne une perte d'information préjudiciable notamment en ce qui concerne les petits mouvements. Par ailleurs, le nombre important de paramètres à régler (6 pour chaque filtre, davantage pour les familles d'ondelettes) est problématique dans l'optique d'une estimation non biaisée et la plus naturelle possible.

2.2.2.2 Corrélation de phase

Les méthodes de corrélation de phase [50] utilisent le domaine de Fourier dont f_x et f_y variables dans l'espace de Fourier, $\hat{f}_1$ et $\hat{f}_2$ étant deux blocs candidats à l'appariement, de même dimension, appartenant aux deux images successives que l'on considère :

$$\hat{f}_2\left(f_x, f_y\right) = \hat{f}_1\left(f_x, f_y\right) e^{-i(f_x.v_x + f_y.v_y)} \tag{2.8}$$

puis

$$\frac{\hat{f}_2.\widehat{f_1^*}}{\left|\hat{f}_2.\widehat{f_1^*}\right|} = e^{-(f_x.v_x + f_y.v_y)} \tag{2.9}$$

La solution du problème est donnée en considérant la surface de corrélation de phase :

$$c_{t,t+1}(v_x, v_y) = F^{(-1)} \frac{\hat{f}_2.\widehat{f_1^*}}{\left|\hat{f}_2.\widehat{f_1^*}\right|} \tag{2.10}$$

Où $F^{(-1)}$ désigne l'opérateur transformé de Fourier inverse. Une estimée du mouvement est alors :

$$[v_x, v_y] = argmax(c_{t,t+1}(v_x, v_y)) \tag{2.11}$$

Cette méthode ne renvoie pas un résultat dense (seuls les maximas locaux sont considérés), et elle donne (dans sa version initiale) des déplacements entiers.

2.2.3 Méthodes de mise en correspondance de blocs Block Matching Algorithm (BMA)

Le principe général de la méthode de mise en correspondance de blocs (Block Matching Algorithm (BMA)) est d'exploiter les redondances temporelles existant entre des images consécutives. Pour cela, considérons une séquence vidéo dans laquelle nous voulons estimer le mouvement de différents objets. Pour simplifier l'estimation, nous ne considérons que le mouvement présent entre deux images successives : l'image courante et l'image de référence. Chaque image est subdivisée en blocs de taille égale (généralement de 8x8 ou 16x16 pixels) et chaque bloc est considéré comme étant un objet indépendant. De manière informelle, l'algorithme consiste, pour un bloc de l'image courante, à choisir un bloc dans l'image de référence (passée ou future) et à calculer un critère comparaison entre ces deux blocs. L'opération est réitérée en choisissant un autre bloc jusqu' à ce que tous les blocs d'une zone déterminée de l'image de référence (appelée ''fenêtre de recherche") aient été testés. Le bloc le plus semblable est ainsi identifié dans l'image de référence pour chaque bloc de l'image courante [51]. Donc BMA estime le mouvement en mesurant un déplacement, obtenu en mettant en correspondance deux blocs issus de deux images de la séquence. Ces méthodes par correspondance de blocs (BMA : Block Matching Algorithm) : sont largement utilisées dans les codeurs vidéo normalisés du fait de leur compromis entre complexité et efficacité de codage, et de leur plus grande facilité d'implantation[52], [53], [54]. C'est pourquoi, nous nous sommes orientés vers ces méthodes. Elles estiment le mouvement en mesurant un déplacement, obtenu en mettant en correspondance deux blocs issus de deux images de la séquence (voir la figure 2.1). Les critères d'appariements les plus utilisés sont la Différence quadratique moyenne (Mean Square Différence MSD), la Différence Absolue Moyenne (Mean Absolute difference MAD), la Somme Absolue des Différences Transformées ou Sum of Absolute Transformed Differences (SATD) [55] et la somme des différences absolue (Sum Absolute Différence) SAD.

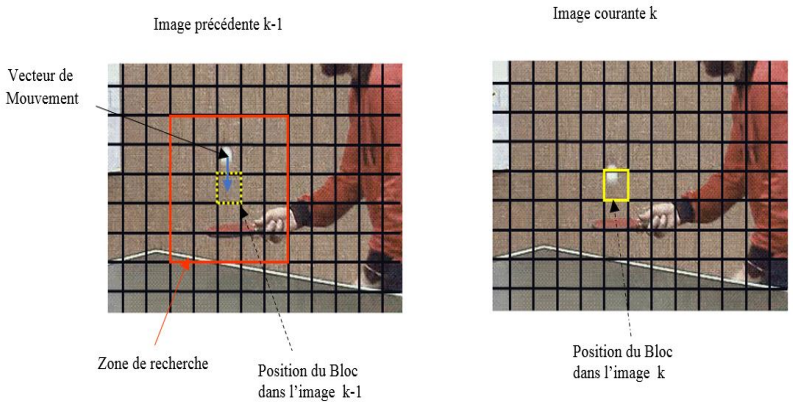


Figure 2.1: Schéma illustratif de l'appariement d'un bloc entre les trames k et k-1.

Avec $f(i,j,k)$ l'intensité du pixel de coordonnées matricielles (i,j) situé dans la trame courante k, et $f(i+dx, j+dy, k-1)$ est l'intensité du pixel correspondant de position $(i+dx, j+dy)$ dans la trame précédente $k-1$, avec $N \times N$ étant la taille du bloc. Après l'application du critère à travers toutes les positions appartenant à la zone de recherche $-x \leq dx \leq +x$ et $-y \leq dy \leq +y$, on sélectionne la valeur minimale de critère de mise en correspondance qui représente le meilleur appariement. C'est-à-dire que le pixel $f(i,j,k)$ de la trame k correspond à $f(i+dx, j+dy, k-1)$ qui est le même pixel, mais il a été déplacé par dx lignes et dy colonnes dans la trame $k-1$. Par conséquent dx et dy représenteront les composantes verticale et horizontale du vecteur de déplacement.

Le critère de la MSD est défini par l'expression suivante :

$$MSD(dx,dy) = \frac{1}{N\times N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[f(i,j,k) - f(i+dx, j+dy, k-1)\right]^2 \tag{2.12}$$

Le critère de la MAD est donné par l'expression suivante :

$$MAD(dx,dy) = \frac{1}{N\times N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|f(i,j,k) - f(i+dx, j+dy, k-1)\right| \tag{2.13}$$

Le critère de la Somme Absolue des Différences Transformées (SATD) est défini par :

$$SATD(dx, dy) = \sum_{i=1}^{N} \sum_{j=1}^{N} H \left| f(i, j, k) - f(i + dx, j + dy, k - 1) \right| \tag{2.14}$$

Avec H la transformée de Hadamard.

Le critère de la SAD est défini par l'expression suivante :

$$SAD(dx, dy) = \sum_{i=1}^{N} \sum_{j=1}^{N} \left| f(i, j, k) - f(i + dx, j + dy, k - 1) \right| \tag{2.15}$$

Pour la plupart des études que nous avons effectuées dans la suite de cette thèse nous avons optés pour l'utilisation de la SAD que nous avons estimé approprié.

2.2.3.1 Algorithme de recherche Full Search (FS)

L'algorithme FS est le plus simple mais aussi le plus coûteux en calcul puisque toutes les comparaisons possibles entre blocs sont effectuées, le bloc de comparaison est alors déplacé pixel par pixel. Le coût en calcul de cet algorithme est en fonction de la taille de l'espace de recherche (fenêtre de recherche). Le bloc retourné par l'algorithme est celui qui minimise le critère de comparaison. On teste tous les blocs de la zone de recherche. Cet algorithme délivre la solution optimale. Pour un bloc donné, la méthode Full Search (FS) calcule toutes les SAD, dans le cas de notre exemple le nombre de SAD calculés dans une fenêtre de recherche pour un bloc est 15x15=225 SAD (ou points testés) comme illustré dans la figure 2.2. La plus petite SAD (SADmin) correspond au vecteur mouvement recherché. Cette recherche exhaustive assure l'optimalité de la solution cependant elle est exigeante en quantité de calculs[39]. Afin de réduire cette complexité plusieurs méthodes ont été proposées. Parmi elles, nous citons : Three Step Search [40], New Tree-Step Search (NTSS) [56], Simple and Efficient Search Algorithm (SES) [57], Four-Step Search (4SS) [58], Diamond Search (DS) [59], Cross Diamond Search (CDS) [60], Hexagon Search (HEX) [61], Octogon Cross Diamond Search (NOCDS)[62], etc. La plus connue est la méthode Diamond Search DS [59]. Qui nous détaillons dans ce qui suit.

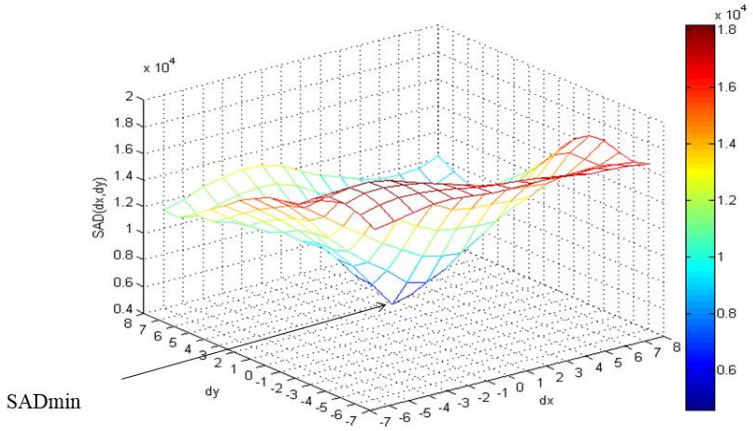


Figure 2.2: Algorithme FS : exemple de forme des l'erreurs SAD obtenus à partir d'une paire d'images de la séquence *News* (bloc de taille 16x16 et une fenêtre de recherche de ± 7 pixels). L'algorithme de FS calcule toutes les SAD c'est a dire 15x15 = 225 SAD pour trouver SADmin.

2.2.3.2 Algorithme de recherche Three Step Search(TSS)

L'algorithme Three Step Search a été proposé en 1981 par Koga et al [40]. Il permet de réduire la complexité de calculs de l'algorithme FS en diminuant le nombre de points testés. La méthode TSS a été adoptée par H.261 et MPEG en raison de sa simplicité et son efficacité. La figure 2.3 illustre la stratégie de recherche de cet algorithme. Dans l'étape 1, les neuf points autour du centre (0,0) sont considérés. Les positions (pour d=4) des huit points sont alors (-4,-4), (0, -4), (+4, -4), (-4,0), (0,0), (+4,0), (-4,+4), (0,+4), et (+4,+4). La position permettant d'obtenir le minimum d'erreur (SADmin) est le centre de la prochaine étape. Dans la deuxième étape, la recherche s'effectue autour du minimum obtenu lors de l'étape précédente, mais avec un pas réduit a moitie. La recherche est réitérée lors de la troisième étape et les huit positions sont ainsi considérées. Lors de cette troisième et dernière phase le pas est à nouveau divisée par deux, les huit nouveaux positions sont testés. Le point ayant le minimum d'erreur est considéré le vecteur de mouvement solution. La figure 2.3 représente un exemple, le bloc initial est positionné à (0,0), neuf points sont calculés lors de l'étape 1. On suppose que (4,0) correspond à la plus petite SAD pour l'étape 1. Pour l'étape 2 (-4,-2) correspond à la plus petite SAD. Enfin après l'étape 3 on trouve que SADmin est située toujours à (-4,-2). Nous limitons la fenêtre de déplacement ±7x7 c'est-à-dire entre (-7, 7) aussi bien pour les abscisses que pour

les ordonnées. Pour cette zone de recherche, l'algorithme de recherche exhaustive FS calcule 15x15=225 SAD alors que TSS calcule seulement 9+8+8=25 SAD. Au lieu de parcourir tous les blocs, le but de ces algorithmes est de parcourir la fenêtre en se rapprochant pas à pas au minimum global.

Le principe de l'algorithme TSS est le suivant :

- **Étape 1 :** Le bloc central et les huit blocs définis à une distance de quatre (d=4 pixels) du pixel central sont testés (voir figure 2.3). Les cordonnées du bloc qui minimise le critère $b_{\min 1}$ sont conservées.
- **Étape 2 :** Même opération que l'étape 1, mais avec $b_{\min 1}$ comme bloc central et d=2, on obtient $b_{\min 2}$.
- **Étape 3 :** Même opération avec d=1 pixel. Les cordonnées du bloc $b_{\min 3}$ représentent le vecteur de mouvement.

La figure 2.3 montre les trois étapes nécessaires à l'algorithme pour retourner un résultat.

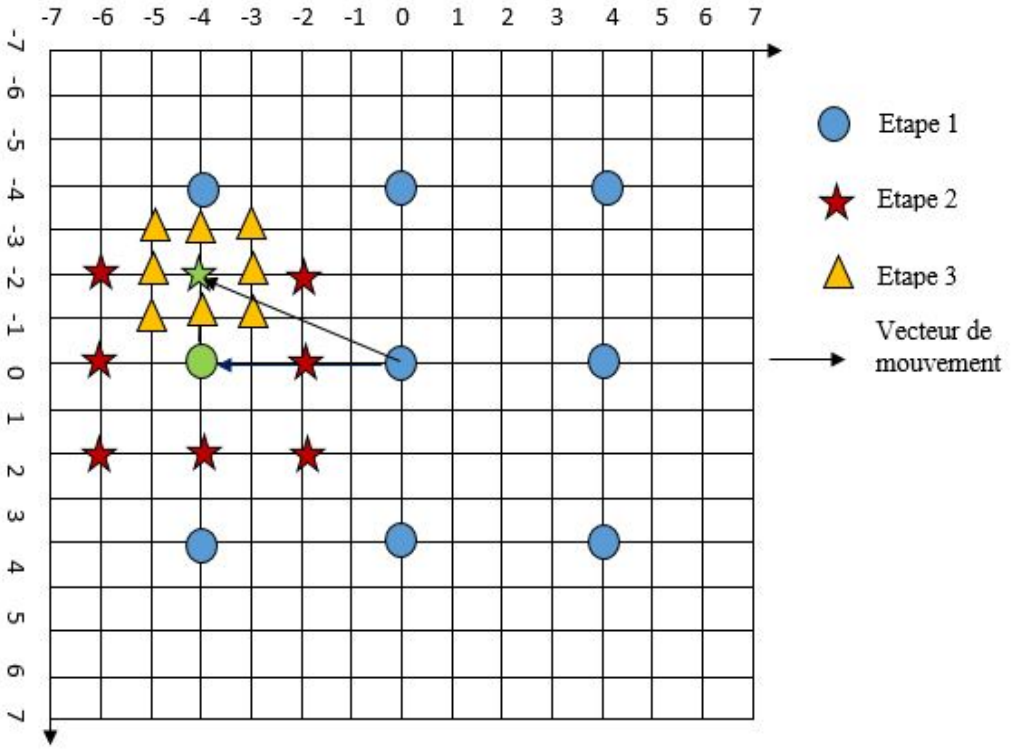


Figure 2.3: Algorithme TSS : exemple de solution se situant à (-4,-2), l'algorithme TSS nécessite 9+8+8=25 points à calculer au lieu de 225 pour FS.

Le nombre des points à tester est ici divisé par 9 par rapport à la méthode de recherche exhaustive FS, ce qui représente une accélération importante en temps de traitement. Le nombre d'étapes de recherche est fixé à 3. Cependant, l'algorithme n'assure pas la solution optimale comme la méthode FS. Cela implique que l'algorithme TSS accepte une dégradation de la qualité d'image, surtout dans les séquences à mouvement lent. Pour remédier à ce problème, l'algorithme d'estimation de mouvement New Three Step Search (NTSS) [56] a été proposé comme solution. Le NTSS se distingue par rapport à TSS par l'ajout de huit nouveaux points pendant la première étape autour du point central, en plus de conditions d'arrêt supplémentaires. L'algorithme NTSS conserve la simplicité et la régularité de la méthode origine TSS, il fonctionne mieux que TSS en termes de qualité d'image compensé, et est tout à fait compatible avec TSS en termes de la complexité de calculs.

2.2.3.3 Algorithme de recherche Four Step Search(4SS)

Cet algorithme (Four Step Search) a été proposé en 1996 par Lai-Man Po et Wing−Chung Ma [58]comme étant un algorithme inspiré de l'algorithme TSS. Celui-ci utilise, comme l'algorithme TSS, un modèle de recherche uniforme en modèle carré. Les deux algorithmes TSS et 4SS utilisent le même motif de recherche en carré. Cependant, il existe deux points de différences majeurs entre ces deux algorithmes. Premièrement, l'algorithme 4SS, la taille du motif est figée pour les premières étapes, et permet contrairement à la méthode TSS d'obtenir des points de chevauchement entre celles-ci, et donc de réduire le nombre des points testés. Deuxièmement, le 4SS offre la possibilité d'un arrêt conditionnel permettant de réduire le nombre total de points testés. La figure 2.4 montre une description générale de fonctionnement du 4SS. La première étape, un modèle de recherche carré de taille 5x5 pixels est utilisé. Il est constitué de neuf points de test. Ce modèle est appliqué pour les trois premières étapes. Cependant, pour l'étape 4, ce modèle est changé pour une taille de 3x3 pixels. Le point ayant l'erreur minimale (SADmin) sera le centre de prochaine étape. Afin de réduire le nombre de points testés, une technique d'arrêt conditionnelle est utilisée par le 4SS. Si le point minimal est celui de centre. L'étape de recherche suivante ira directement au modèle de taille 3x3 de sorte que le nombre de points de test sont réduits, Le nombre de points testés est réduit dans les meilleures cas a 17 (9+8=17)

(voir figure 2.5).

L'algorithme 4SS se résume de la manière suivante :

- **Étape 1 :** Le bloc central et les huit blocs définis à une distance (d=2 pixels) (voir figure 2.4) sont testés. Si le bloc $b_{\min 1}$ qui minimise la SAD est au centre alors aller à l'étape 4, sinon continue.
- **Étape 2 :** Le bloc $b_{\min 1}$ est le centre et d=2, les huit points sont calculés. Si $b_{\min 2}$ qui minimise la SAD est le centre, allez à l'étape 4 sinon continue.
- **Étape 3 :** $b_{\min 2}$ est le centre et d=2, les huit points sont calculés. Si le point $b_{\min 3}$ qui minimise la SAD est au centre, allez à l'étape 4.
- **Étape 4 :** d=1 les huit points autour du centre de la recherche sont examinés. Le point obtenu (minimisant l'erreur) représente la solution finale.

Exemples d'un chemin de recherche en quatre étapes est présentés sur la figure 2.4.

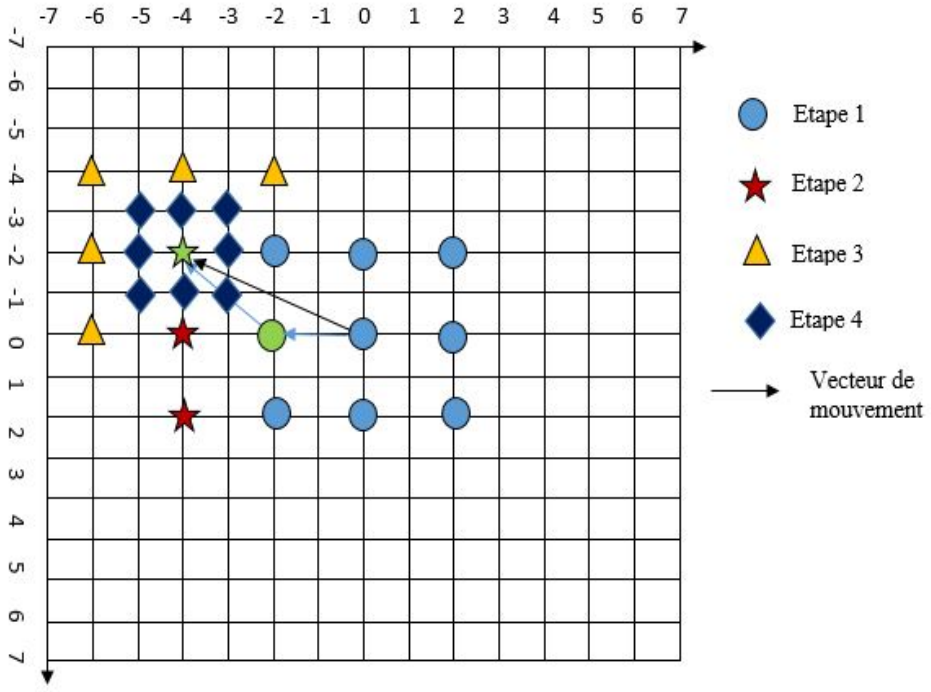
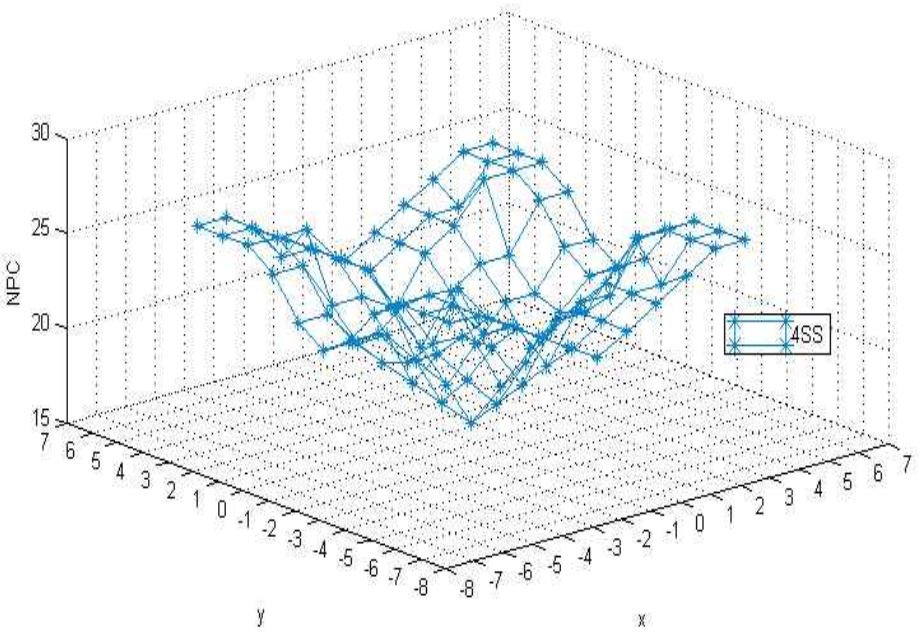
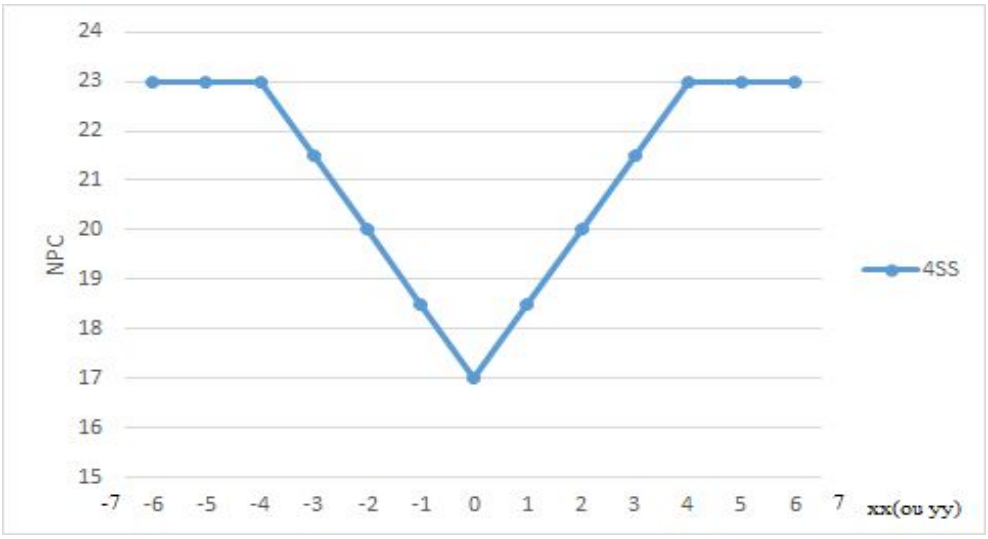


Figure 2.4: Algorithme 4SS : exemple de solution se situant à (-4,-2), l'algorithme 4SS nécessite le calcule de 9+3+5+8= 25 SAD.



(a) Algorithme 4SS : NPC en fonction du déplacement dans la fenêtre de recherche (-7, 7).



(b) Algorithme 4SS : NPC sur l'axe xx ou yy.

Figure 2.5: Algorithme 4SS : NPC en fonction du déplacement dans la fenêtre de recherche [-7, 7] en (a) et sur l'axe des xx (ou yy) en (b).

En outre, à chaque étape, il existe plusieurs points de recouvrement. Selon la position de l'erreur minimale, c'est seulement 3 ou 5 points qui demeurent à tester. Ainsi, lors des 4 étapes consécutives, le nombre maximum de points testés sera donc égal à 27 (9+5+5+8). Le nombre

de points varie entre un minimum de 17 et un maximum de 27. Bien que cela dépende de la nature de l'image, le nombre de points testés en moyenne généralement est inférieur à celui de la méthode TSS.

2.2.3.4 Algorithme de recherche Diamond Search(DS)

L'algorithme Diamond Search (DS) [59] ressemble à l'algorithme 4SS. La différence c'est que le modèle de recherche carré est changé en diamant. L'algorithme DS utilise deux schémas de recherche illustrés dans la figure 2.7. Les points des modèles de recherche employés sont dérivés des travaux statistiques effectués sur la distribution de la fréquence du vecteur de mouvement en fonction de son amplitude qui montrent que celui-ci est à 80% des cas à l'intérieur d'un disque de rayon égal à 2 pixels. Les 13 points montrent tous les points de test possibles dans le disque de rayon 2 pixels (voir la figure 2.6) [59].

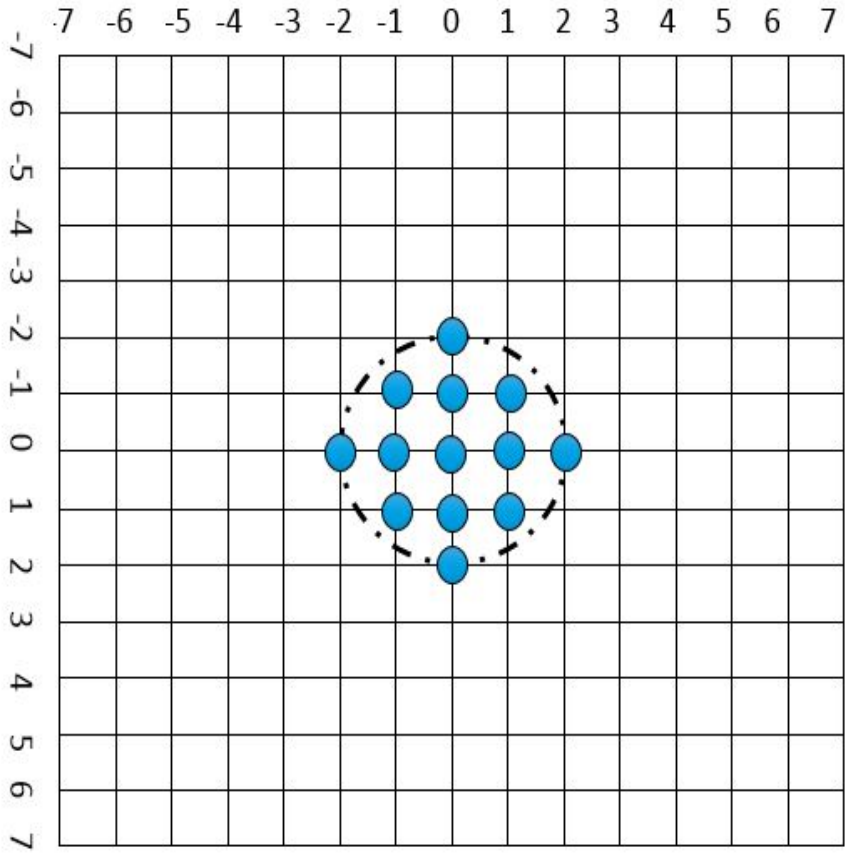
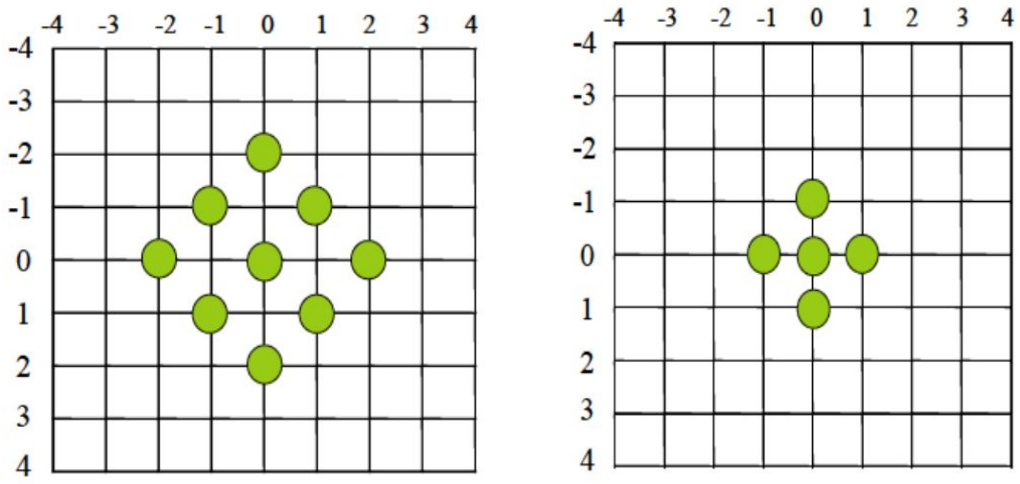


Figure 2.6: Schéma de recherche de DS : disque de rayon de 2 pixels. Tous les points de test possibles dans le disque de rayon 2 pixels sont représentés (13 points), ils représentent 80% des cas possibles de déplacement.

Le premier modèle (voir 2.7) appelé Large Diamond Search Pattern (LDSP) en (a) est constitué de neuf points dont huit points à une distance de deux pixels par rapport au centre. Le neuvième point est situé au centre pour composer une forme de diamant. Le Second modèle nommé Small Diamond Search Pattern (SDSP) en (b) contient cinq points dont quatre sont sur le bord situés à une distance d'un pixel par rapport au centre. Le cinquième est situé au centre.



(a) LDSP (b) SDSP

Figure 2.7: Les différents modèles de recherche utilisés dans l'algorithme Diamant ; Large Diamond Search Pattern (a) suivi par Small Diamond Search Pattern en (b).

La figure 2.8 illustre le déroulement de l'algorithme dans les trois directions de la recherche.Les modèles LDSP se recouvrent partiellement. Comme nous l'expliquons par la suite, le nombre de points calculés par l'algorithme est directement lié a la complexité de celui ci, c'est pour cela que nous présentons dans ce qui suit une étude des différents cas de déplacements. Pour l'étude de la complexité de DS. On considère un exemple simple. DS commence par le schéma LDSP et donc 9 points sont calculés (figure 2.7 (a)). Trois cas peuvent se présenter :

– Le minimum se situe diagonalement par rapport au schéma LDSP (voir figure 2.8 (a)). Dans ce cas, l'algorithme continue en calculant trois autres points.

- Le minimum se situe (voir figure 2.8 (b)) sur un des sommets du schéma LDSP : l'algorithme continue en calculant cette fois-ci 5 points.
- Le minimum se situe au centre (figure 2.8 (c)). L'algorithme utilise cette fois-ci le schéma SDSP et 4 points sont alors calculés.

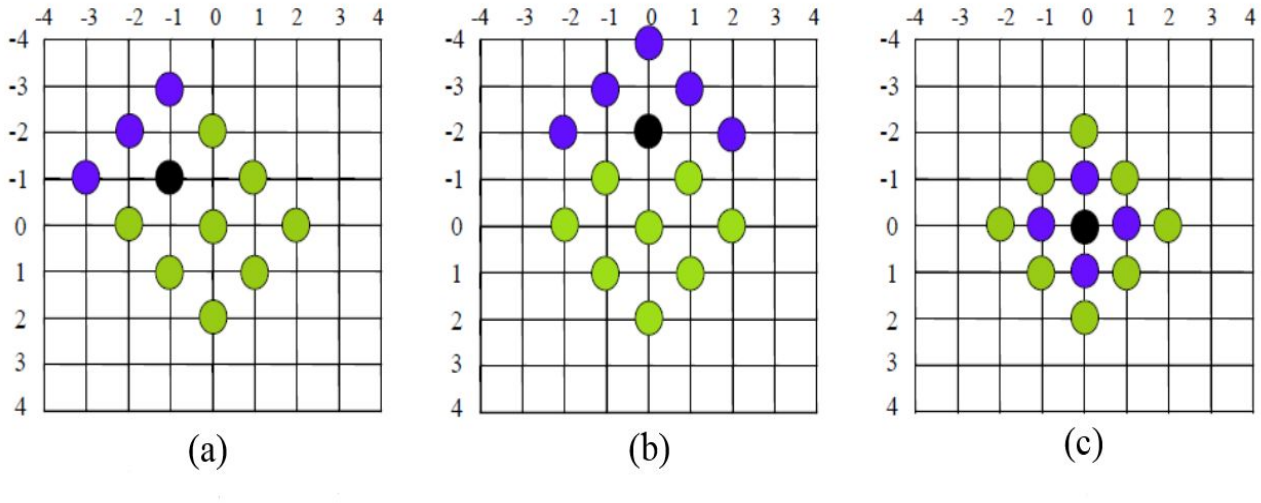


Figure 2.8: Algorithme DS : 3 cas de déplacement possible : diagonalement en (a) horizontalement ou verticalement en (b), le minimum est au centre (c). On remarque que 3, 5 et 4 points sont respectivement calculés.

L'algorithme DS :

- **Étape 1 :** Le LDSP est centré sur l'origine du point courant. Neuf points sont testés. Si le bloc $b_{\min}$ qui minimise l'erreur correspond au centre de LDSP, on va à l'étape 3, sinon, on continue.
- **Étape 2 :** le bloc $b_{\min}$ est considéré comme le centre d'un nouveau LDSP. Après calcul si le nouveau $b_{\min}$ est au centre du LDSP alors aller à l'étape 3. Si non, on répète l'étape 2.
- **Étape 3 :**Le schéma LDSP est changé par le schéma SDSP et les quatre points sont testés. Le point obtenu lors de cette étape est la solution finale.

Le nombre de points varie donc entre un minimum de 13 pour les blocs statiques et un maximum conditionné par la troisième étape. Bien que cela dépend de la nature de l'image, le nombre de points testés en moyenne est inférieurs à celui de la méthode 4SS (voir la figure 2.10).

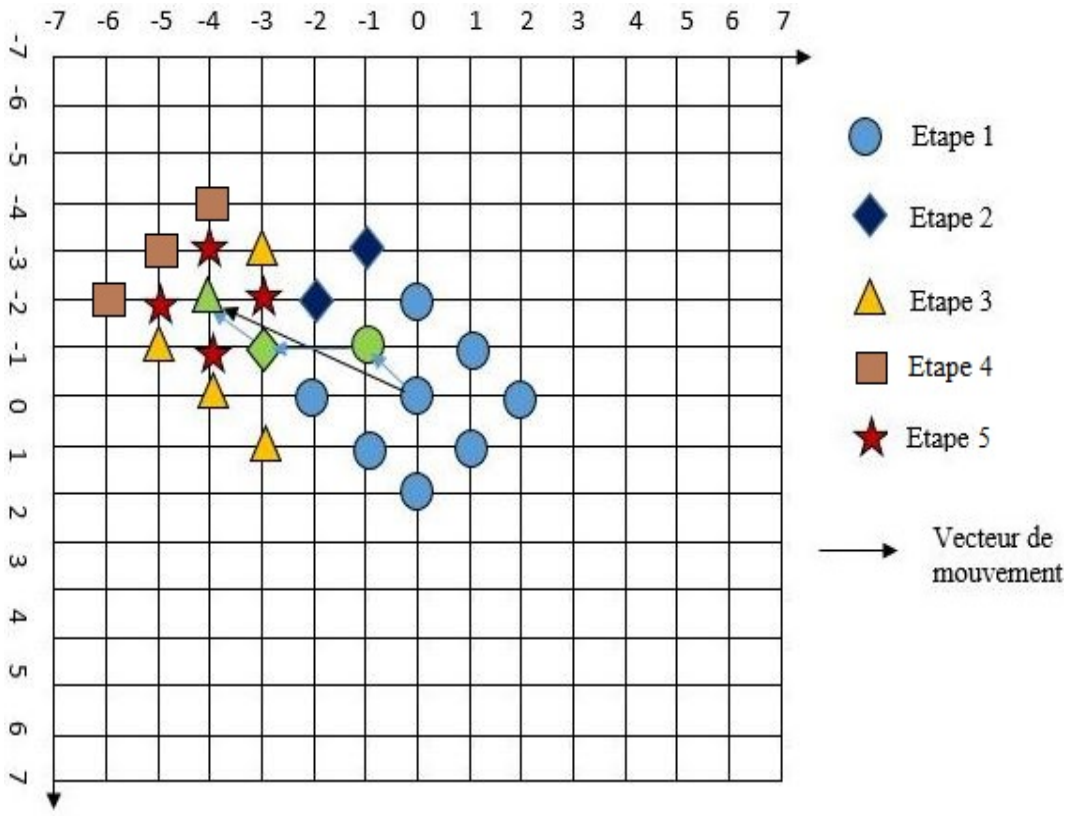
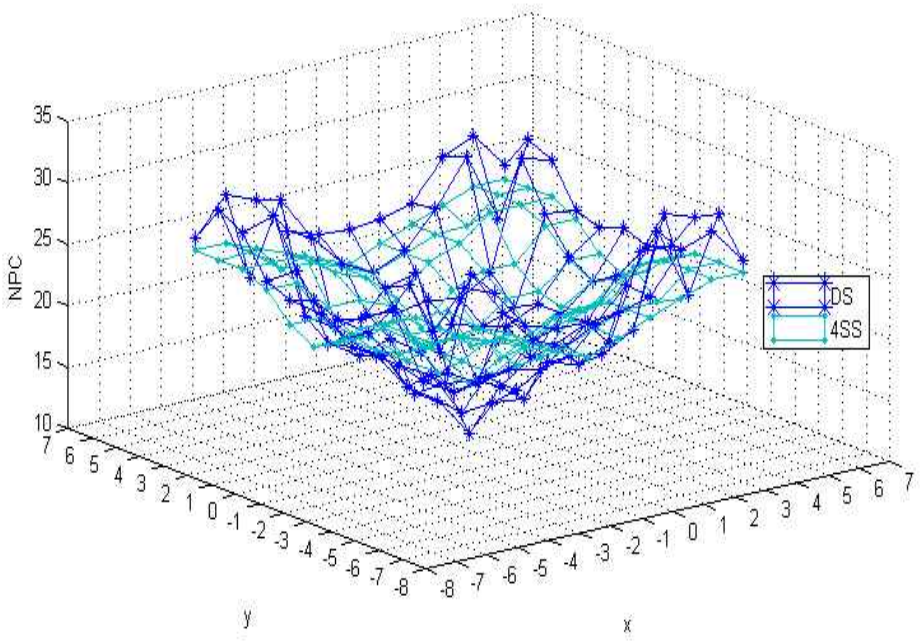
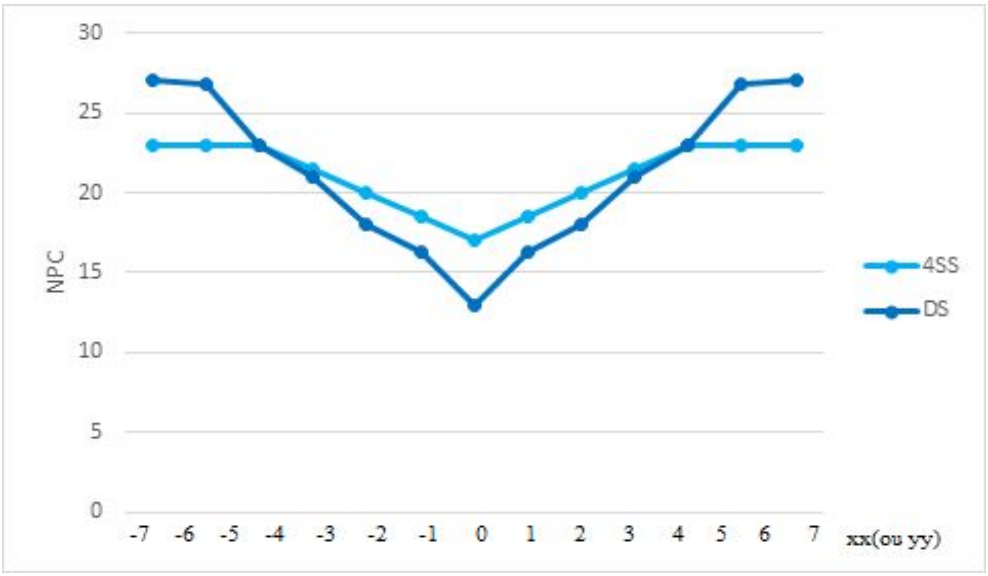


Figure 2.9: Algorithme DS : exemple de solution se situant à (-4,-2) , l'algorithme DS effectue quatre recherches LDSP et une SDSP comme illustré dans la figure. Cela nécessite 9+3+5+3+4 = 24 points à calculer.

L'algorithme DS permet, comme nous le verrons dans la suite de cette thèse, d'accélérer la recherche dans le sens ou moins de points sont testés par rapport aux TSS, NTSS, 4SS tout en augmentant la qualité de l'estimation.



(a) NPC sur toute la fenêtre de recherche (-7, 7) calculé par les algorithmes DS et 4SS.



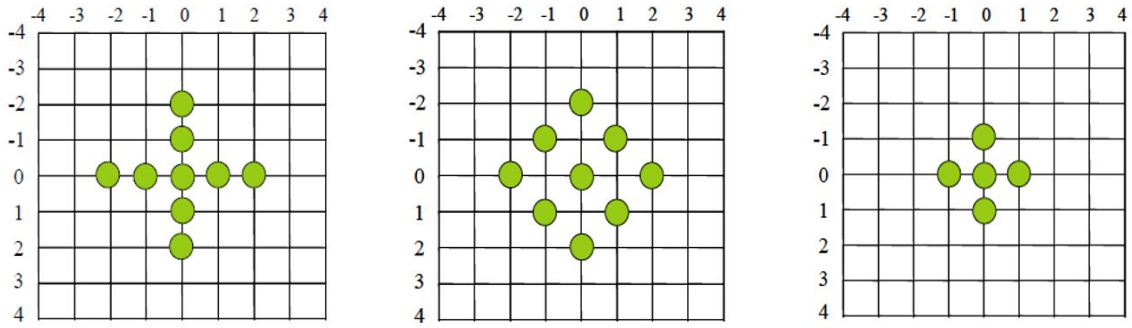
(b) NPC sur l'axe xx ou yy calculé par l'algorithmes DS et 4SS.

Figure 2.10: NPC dans la fenêtre de recherche [-7, 7] en (a) et sur l'axe des xx (ou yy) en (b) calculé par les algorithmes DS et 4SS.

On remarque figure 2.10 que DS est moins complexe que 4SS pour des déplacements inférieur à 4.

2.2.3.5 Algorithme de recherche Cross-Diamond Search (CDS)

La méthode de Cross Diamond Search (CDS) [60] est une alternative de l'algorithme Diamond Search. Celle-ci utilise trois Schémas de recherche illustrés par la figure 2.11. Le premier schéma appelé Cross Search Pattern (CSP) composé neuf points dont le neuvième point est situé au centre de la Croix. Le deuxième schéma appelé Large Diamond Search Pattern (LDSP) comprend neuf points dont huit points sur le bord du Diamant à une distance de deux pixels par rapport au centre. Le neuvième point est situé au centre pour composer une forme de Diamant. Le troisième schéma nommé Small Diamond Search Pattern (SDSP) contient cinq points dont quatre sont sur le bord situés à une distance d'un pixel par rapport au centre. Le cinquième est situé au centre. Cela permet d'entrevoir une qualité d'image approximativement équivalente de celle de DS avec un nombre de points supérieur (complexité plus par rapport au DS)(voir la figure 2.13).



(a) CSP (b) LDSP (c) SDSP

Figure 2.11: Les différents modèles de recherche utilisés dans l'algorithme Cross-Diamond Search ; Cross Search Pattern en (a), Large Diamond Search Pattern en (b) et Small Diamond Search Pattern en (c).

L'algorithme de recherche en CDS :

- **Étape 1 :** Le CSP est centré sur l'origine du point courant et les neuf points sont testés. Si le point qui minimise la SAD est celui du centre, alors cette étape représente la solution finale pour obtenir le vecteur mouvement du point. Sinon, on continue.
- **Étape 2 :** Quatre points sont ajoutés au schéma CSP pour former un LDSP qui est centré sur l'origine du point courant et les quatre points sont testés. Le point minimisant la SAD est choisi le centre d'un nouveau LDSP pour la troisième étape.
- **Étape 3 :** L'origine du point minimisant la SAD calculée précédemment est repositionnée pour être le centre d'un nouveau LDSP. Les points du schéma sont alors testés successivement. Si le nouveau point minimisant la SAD est le point central,continue avec la quatrième étape. Si non, répéter la troisième étape.
- **Étape 4 :** Le schéma LDSP est changé avec le schéma SDSP et les points sont testés. Le point obtenu lors de cette étape représente la solution finale.

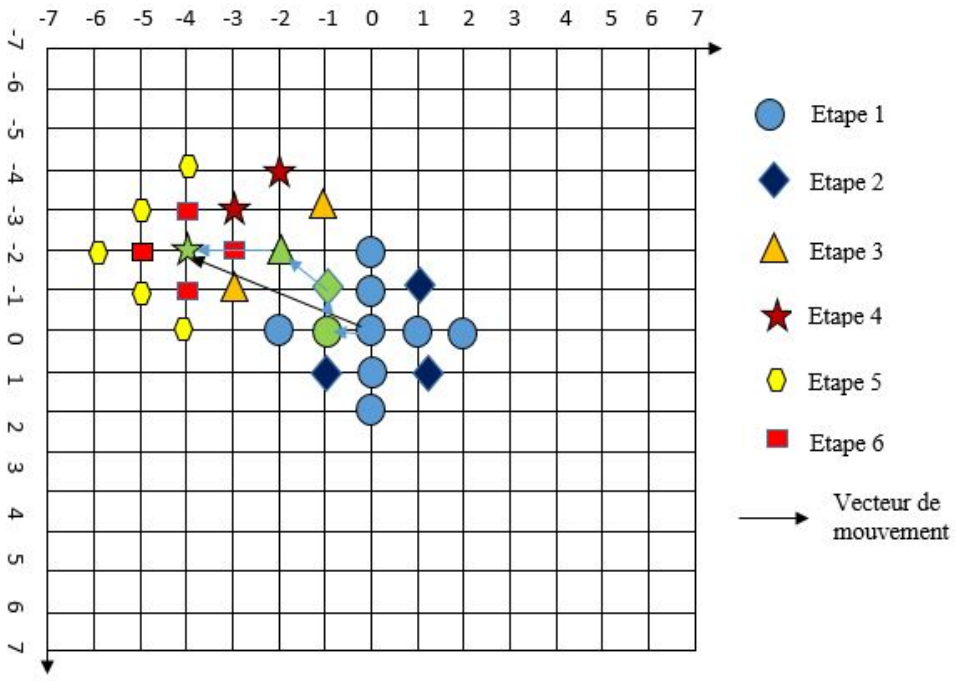
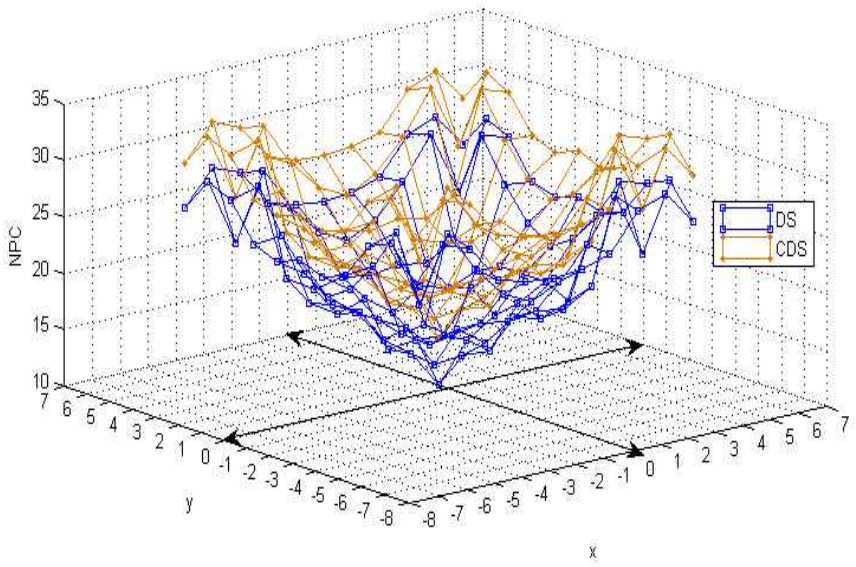


Figure 2.12: Algorithme CDS : exemple de solution se situant à (-4,-2) : CDS effectue une recherche CSP, 4 points, après deux(LDSP) et une SDSP comme illustré dans la figure. Cela nécessite 9+4+3+3+5+4= 28 points à calculer.

La figure 2.12 illustre le déroulement de l'algorithme en indiquant la différence entre les quatre étapes. On constate qu'à partir de la deuxième étape l'algorithme devient similaire au DS.



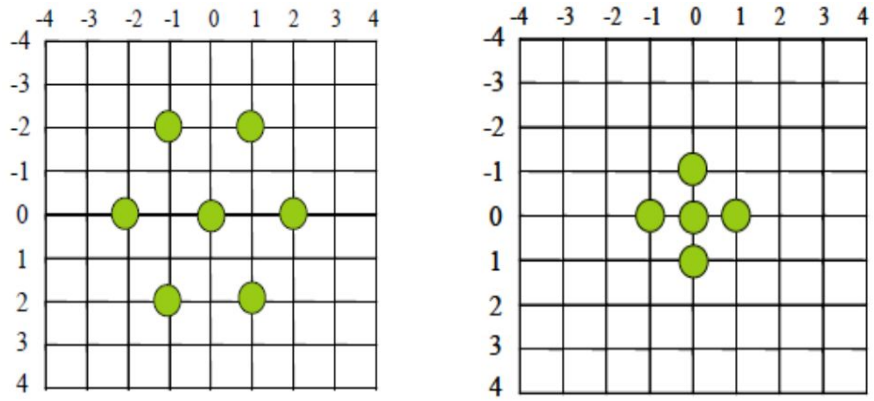
(a) NPC sur toute la fenêtre de recherche (-7, 7) calculé par les algorithmes CDS et DS.

(b) NPC sur l'axe xx ou yy calculé par les algorithmes CDS et DS.

Figure 2.13: NPC dans la fenêtre de recherche [-7, 7] en (a) et sur l'axe des xx (ou yy) en (b) calculé par les algorithmes CDS et DS. On remarque que DS est moins complexe que CDS.

2.2.3.6 Algorithme de recherche en hexagone Hexagonal Search (HEX)

L'algorithme Hexagonal Search (HEX) est une variante de l'algorithme DS [61]. En effet, l'algorithme utilise, comme dans le DS, deux schémas de recherche illustrés dans la figure 2.14 : un schéma large (Large Hexagonal Search Pattern : LHSP) et un schéma petit (Small Hexagonal Search Pattern : SHSP). La première remarque que l'on peut faire est que le SHSP est identique au SDSP. Deuxièmement, le LHSP contient sept points alors que le LDSP en contient neuf. Ceci représente un avantage par rapport à l'algorithme DS au niveau de l'implémentation car il n'y a pas de cas particulier, tout est géré identiquement et nous sommes certains que chaque itération de la deuxième étape n'ajoute que le test de trois points. Cela permet d'entrevoir la diminution du nombre de points testés. Cependant, la qualité d'image est moins par rapport à Diamond, de fait qu'elle ignore les déplacements purement verticaux et diagonaux(voir les positions des points de schéma LHSP de la figure 2.14 (a)).



(a) LHSP (b) SHSP

Figure 2.14: Les différents modèles de recherche utilisés dans l'algorithme Hexagonal Search ; Large Hexagonal Search Pattern en (a) suivi par Small Hexagonal Search Pattern en (b).

Nous pouvons voir sur la figure 2.15 que, quel que soit l'emplacement du point minimisant le critère, seuls trois blocs devront être testés pour l'itération suivante. Ceci représente un avantage par rapport à l'algorithme DS au niveau de l'implémentation car il n'y a pas de cas particulier, tout est géré identiquement.

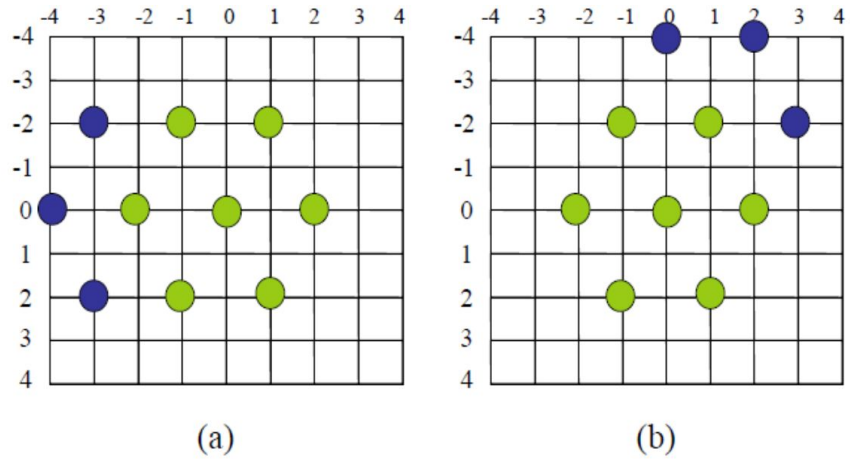


Figure 2.15: Algorithme HEX : les cas (a)(b) correspondent respectivement à SADmin au point (horizontal ou vertical) et sur la diagonale.

L'algorithme HEX, dont un exemple de déroulement est donné par la figure 2.16, est similaire au DS mis à part l'utilisation des modèles de recherche LHSP et SHSP à la place des modèles LDSP et SDSP. Enfin, nous pouvons noter que l'hexagone permet de diminuer le nombre de points de test par rapport au l'algorithme DS(voir la figure 2.17). Cependant, les tests mettent en évidence le fait qu'il est en général moins efficace au niveau qualitatif [25](le minimum trouvé par HEX est souvent éloigné du minimum réel).

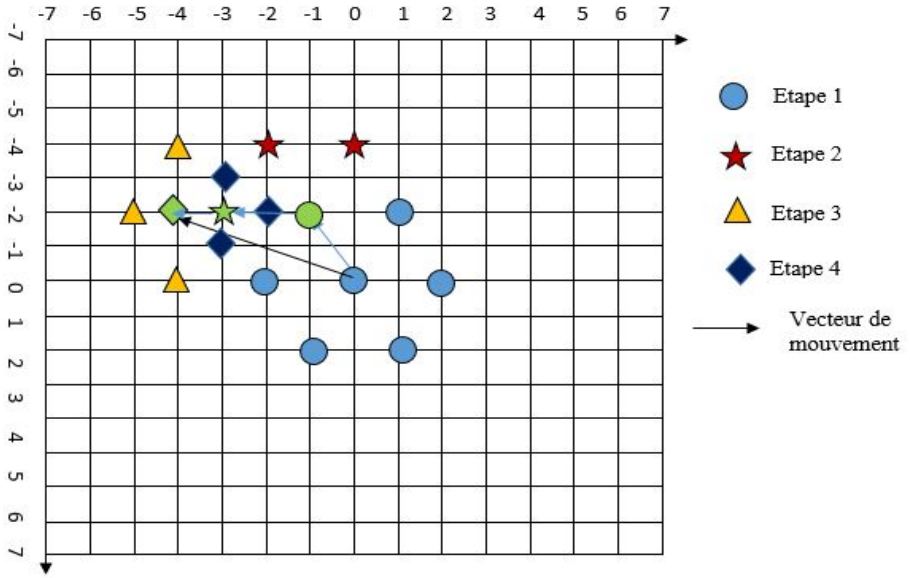
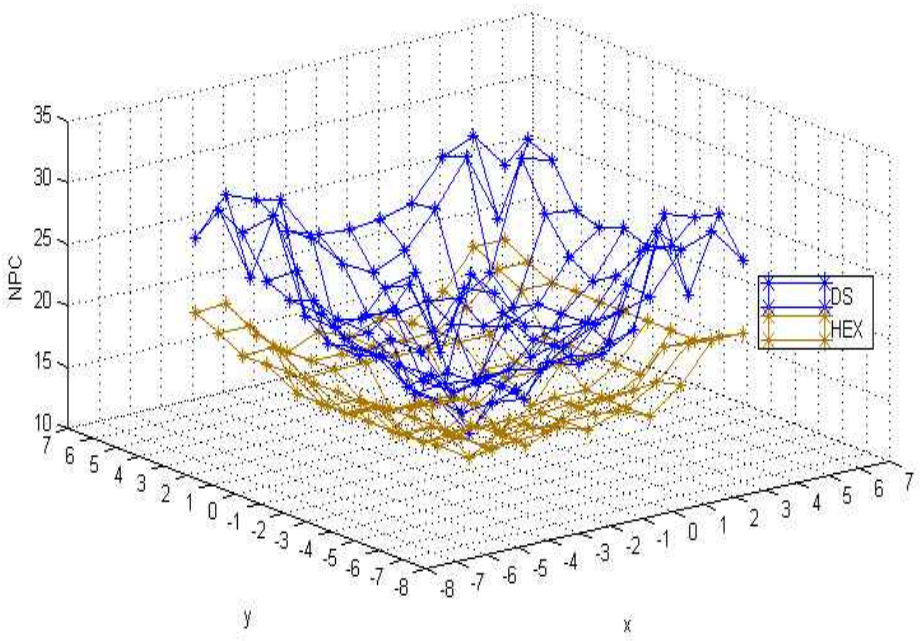
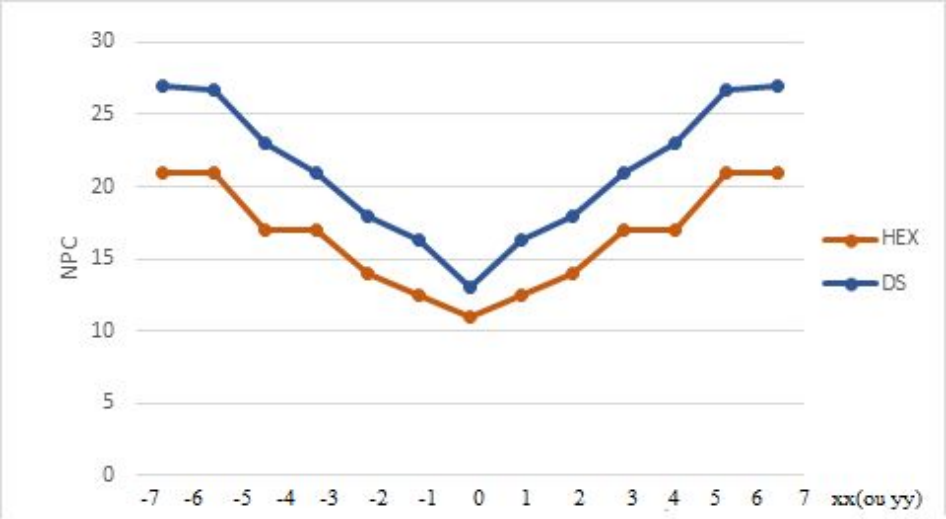


Figure 2.16: Algorithme HEX : exemple de solution se situant à (-4,-2) : HEX effectue trois recherches LHSP et une recherche SHSP comme illustré dans la figure. Cela nécessite 7+3+3+4= 17 points à calculer.



(a) NPC sur toute la fenêtre de recherche (-7, 7) calculé par les algorithmes HEX et DS.



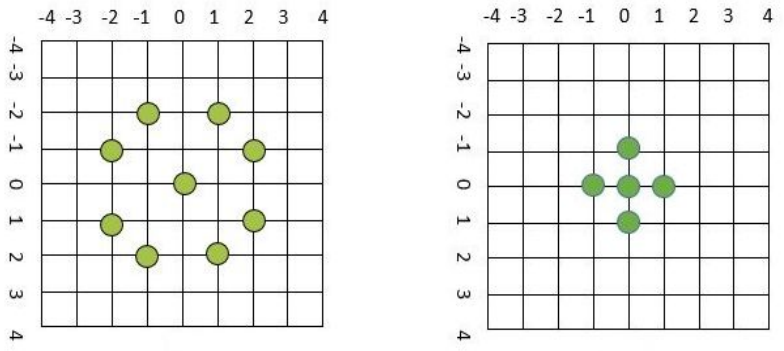
(b) NPC sur l'axe xx calculé par les algorithmes HEX et DS.

Figure 2.17: NPC dans la fenêtre de recherche [-7, 7] en (a) et sur l'axe des xx (ou yy) en (b) calculé par les algorithmes HEX et DS.

2.2.3.7 Algorithme de recherche Novel Octogon Cross Diamond Search (NOCDS)

L'algorithme NOCDS utilise deux schémas de recherche, le schéma octogone suivi par le modèle Small Diamond Search Pattern (voir la figure 2.18)[62]. La recherche s'arrête si l'erreur

minimale se trouve au centre de modèle Small Diamond Search Pattern. Les points des schémas employés par NOCDS par rapport à ceux utilisés par DS sont appropriés et enferment le disque de rayon égal à 2 pixels (voir la figure 2.6). Cependant, NOCDS permet une qualité d'image moins de celle de DS de fait qu'elle ignore les déplacements purement verticaux et horizontaux (voir les positions des points de schéma de la figure 2.18 (a)). Le nombre de points calculés équivalents au DS sur xx ou yy et supérieur par ailleurs(-7, 7) (voir la figure 2.20).



(a) Octogon (b) SDSP

Figure 2.18: Les différents modèles de recherche utilisés dans l'algorithme NOCDS ; Octagon Search Pattern en (a) suivi par Small Diamond Search Pattern en (b).

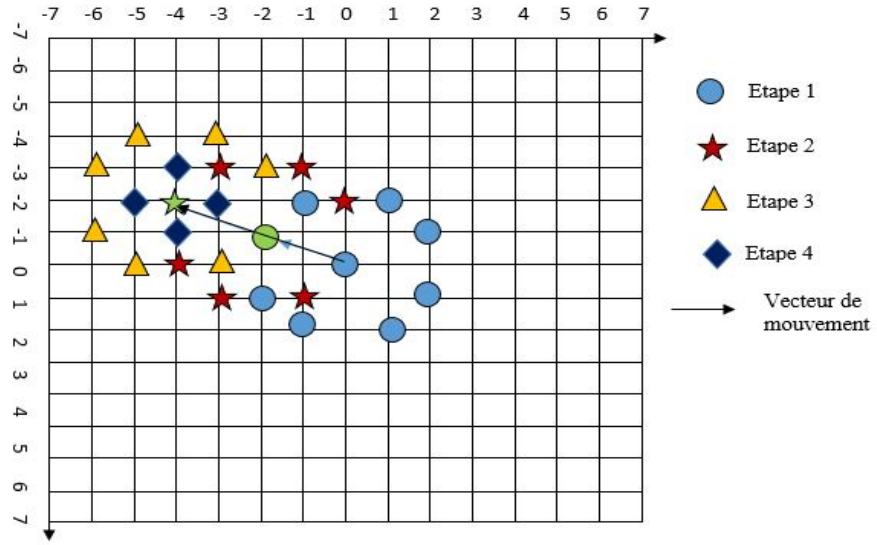
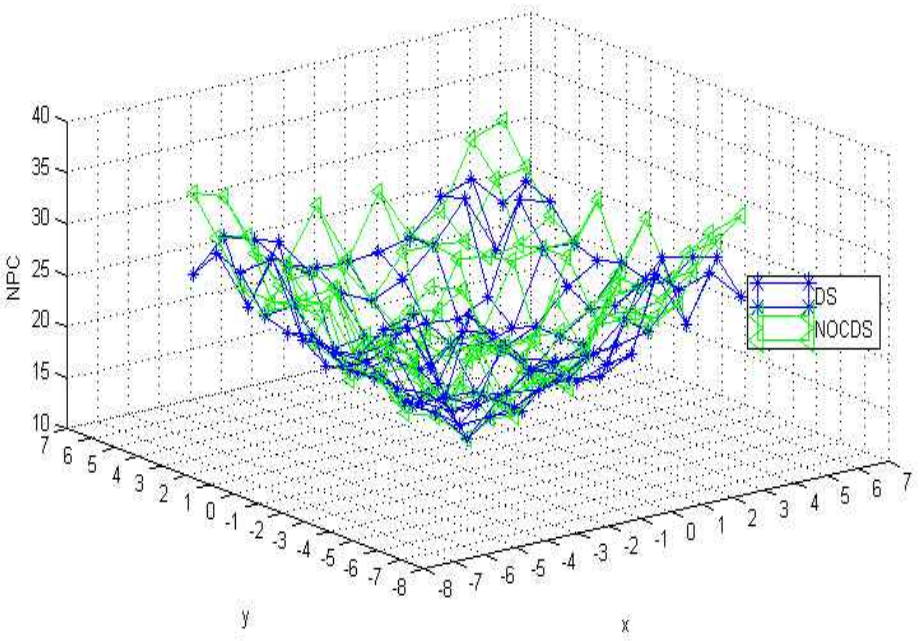
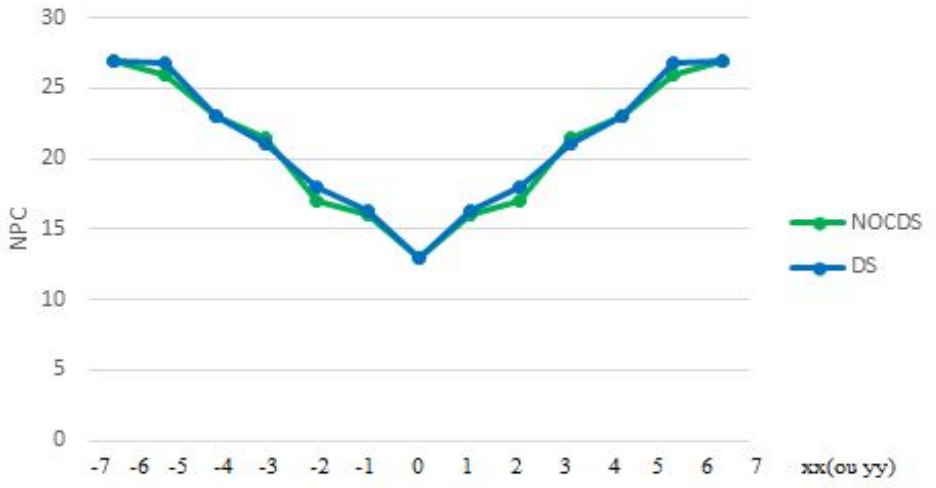


Figure 2.19: Algorithme NOCDS : exemple de solution se situant à (-4,-2) : NOCDS effectue trois recherches octogones et une recherche petite diamant comme illustré dans la figure. Cela nécessite 9+7+7+4= 27 points à calculer.



(a) NPC sur toute la fenêtre de recherche (-7, 7) calculé par les algorithmes NOCDS et DS.



(b) NPC sur l'axe xx ou yy calculé par les algorithmes NOCDS et DS.

Figure 2.20: NPC dans la fenêtre de recherche [-7, 7] en (a) et sur l'axe des xx (ou yy) en (b) calculé par les algorithmes NOCDS et DS.

2.3 Discussion

Nous avons étudié la complexité des algorithmes DS, NOCDS, 4SS, CDS et TSS à l'aide de paramètre NPC. La figure 2.21 donne une vision 3D du paramètre NPC de toutes ses méthodes. Afin de mieux discerner entre les différents courbes nous présentons les coups xx, yy et xy de celle ci (voir figure 2.22 et 2.23).

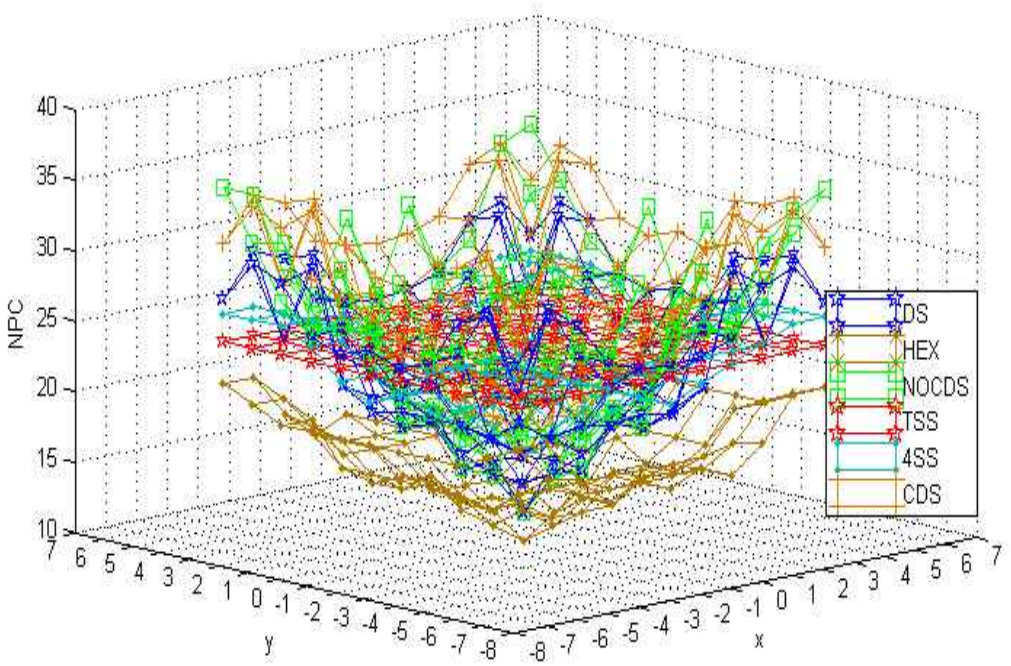


Figure 2.21: NPC dans la fenêtre de recherche [-7, 7] calculé par les algorithmes TSS, 4SS, DS, HEX, CDS et NOCDS.

L'algorithme HEX bien qu'il soit plus rapide (de moindre complexité) (voir figures 2.21, 2.22 et 2.23). La qualité des séquences obtenues sont médiocres d'une manière générale. Les séquences obtenues par HEX sont "visuellement" dégradées comparativement aux méthodes DS, NOCDS, CDS, 4SS et TSS qui sont de qualité équivalente.

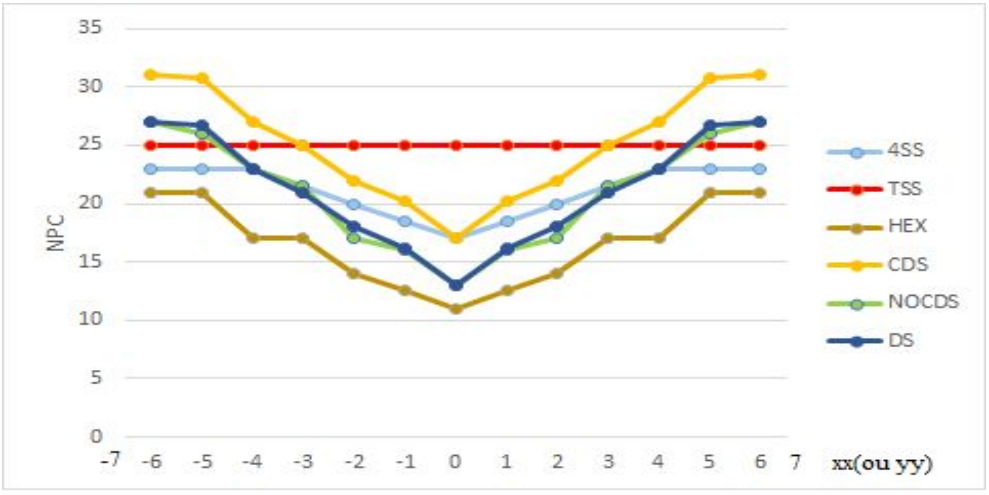


Figure 2.22: NPC sur l'axe xx (ou yy) calculé par les algorithmes TSS, 4SS, DS, HEX, CDS et NOCDS.

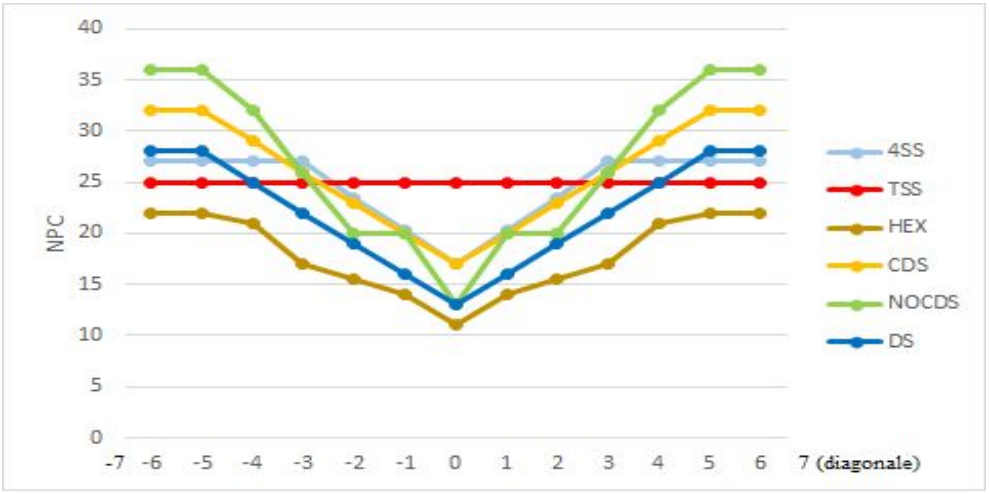


Figure 2.23: NPC sur la diagonale calculé par les algorithmes TSS, 4SS, DS, HEX, CDS et NOCDS.

(a) FS : *PSNR=23.78 dB* (b) DS : *PSNR=20.31 dB* (c) HEX : *PSNR=20.22 dB*

Figure 2.24: Algorithmes FS, DS et HEX : la qualité de l'image reconstruite.

La figure 2.24 donne un exemple à appréciation de la qualité obtenue par HEX et DS pour la séquence *Bus*. Nous nous somme intéressé surtout aux méthodes DS, NOCDS, 4SS, CDS et TSS. Les méthodes TSS, CDS et 4SS comparativement à DS sont plus complexes (pour une même qualité). Il en ressort que DS et NOCDS (pour une qualité pratiquement identique), ils sont les moins complexes . Une étude plus fine (voir figure 2.23) montre que d'une manière globale DS est meilleurs. Cela explique pourquoi nous avons choisi pour l'étude qui suit la méthode DS comme méthode référence.

2.4 Conclusion

Nous avons présenté et étudié les différentes techniques qui permettent l'estimation du mouvement. Parmi ces techniques, nous nous sommes intéressés principalement aux méthodes de type « Block Matching Algorithm ». Cette technique comporte une grande variété algorithmique permettant de calculer la similarité entre les blocs. Nous avons pu relever que chaque méthode possède ses propres avantages et inconvénients en termes de complexité algorithmique et qualité d'image résultante. Une étude comparative a été présentée se basant sur la qualité/complexité. Nous avons proposé nouveau paramètre que nous avons nommé NPC (Nombre de Points Calculés) comme outil permettant de mesurer la complexité de l'algorithme. Nous avons trouvé que DS est le meilleur algorithme de point de vue complexité (pour une même qualité de l'image).

CHAPITRE 3

ESTIMATION DU MOUVEMENT : MÉTHODES PROPOSÉES

3.1 Introduction

L'estimation de mouvement (EM) est un domaine essentiel pour la compression de données (données multi-média mobile 3G et 4G). Elle supprime la redondance temporelle entre les images adjacentes de la séquence vidéo et donc permet un codage efficace. Les algorithmes d'estimation de mouvement par la mise en correspondance de blocs sont largement adoptés par la plupart des standards de codage vidéo, tel que H.264/AVC et MPEG4/AVC [55] grâce à leur bonne compression et leur implémentation simple.

L'algorithme de recherche exhaustive (FS), consiste à diviser l'image courante en blocs et chercher le meilleur appariement pour chacun à l'intérieur d'une fenêtre de recherche dans l'image de référence. Le vecteur de mouvement est obtenu après une recherche exhaustive de tous les cas possibles dans une fenêtre de recherche [39]. Cependant l'algorithme FS est d'une grande complexité. Il est incompatible aux applications en temps réel [63]. C'est pour cette raison que beaucoup d'algorithmes de recherche rapide ont été proposés [40, 56, 57, 58, 59, 60, 61, 62, 41]. Le principe des stratégies d'accélération est de parcourir « intelligemment » l'espace de re-

cherche pour minimiser la complexité de calcul d'un bon d'appariement. Le balayage non exhaustif orienté consiste à exploiter différents modèles et stratégies de recherche pour minimiser la complexité de calcul d'un bon appariement. D'une manière générale, il consiste à explorer le voisinage direct de la fenêtre de recherche.

Ce chapitre est organisé en deux parties correspondant à chacune de nos contributions. La première contribution est le nouvel algorithme que nous avons nommé par Étoile-Diamant (ED). Celui-ci, comme Diamond Search (DS) est élaboré à partir d'une étude statistique. Les auteurs de DS (algorithme le plus utilisé actuellement) partent à partir du constat que 80% des vecteurs de mouvement se situent dans un disque rayon 2 pixels. Cependant, aucune étude quantitative et statistique n'a été effectuée pour quantifier, d'une manière plus fine, le comportement du vecteur de mouvement sur tout le long des axes xx et yy et d'une manière générale sur tout le plan xy. Pour cela nous avons effectué une étude statistique sur un corpus de 12 séquences d'images. Elle confirme les résultats obtenus par les auteurs de DS mais aussi affinent les résultats : on observe une forte concentration des vecteurs de mouvement dans les axes xx et yy. C'est sur cette base qu'on a construit l'algorithme Étoile-Diamant. Il exploite cette observation et possède une complexité réduite pour les vecteurs de mouvement de rayon supérieur à 2 pixels.

La deuxième contribution : un algorithme qui permet d'accélérer l'estimation du mouvement à l'aide d'un seuil. Ceci est basé sur une observation simple : très souvent le champ de vecteurs de mouvement est majoritairement rempli de zéros (mouvement nul). L'idée a consisté à réduire l'ensemble lors de la recherche de meilleur appariement des blocs à ausculter à l'aide d'un seuil approprié. Une étude comparative avec l'ensemble des méthodes existants les plus connues dans la littérateur est effectuée montrant le bien fondé de deux méthodes proposées.

3.2 Distribution des vecteurs de mouvement dans les images réelles.

Les travaux statistiques effectués sur la distribution de la fréquence des vecteurs de mouvement en fonction de l'amplitude montrent que celui-ci est à 80% des cas à l'intérieur d'un disque de rayon égal à 2 pixels [59]. Ceux-ci affirment aussi que sa fréquence est relativement

élevée sur les axes ; horizontal xx et vertical yy sans donner aucune information quantitative. Afin d'élargir notre connaissance sur sa distribution nous avons effectué une étude de la distribution de la fréquence du vecteur mouvement sur un corpus de 12 vidéos composées chacune de 150 images (352x288 pixels par image) [64]. Ce corpus couramment employé (voir figure 3.1) contient aussi bien des séquences d'images à mouvement lent comme *Silent*, *Paris* et *Container* ou des séquences à mouvement rapide comme *Bus*, *Stefan* et *Football* ainsi que les séquences avec un mouvement du caméra horizontal tel que *Flower*. Ce corpus est représentatif des vidéos que l'on peut trouver en réalité. En effet, par exemple la vidéo de *Foreman* représente un personnage centré sur la scène avec un fond fixe et seul le visage bouge. La vidéo *Flower* est une translation latérale de caméra filmant une scène où les différents objets sont placés à différentes profondeurs. La conséquence de cette translation est que les différents objets ont des mouvements apparents différents : rapides pour les objets proches et quasi−nul, pour les objets lointains. De plus, cette vidéo est particulièrement intéressante dans la mesure où elle contient des objets extrêmement texturés (e.g. les fleurs) ainsi que des objets homogènes (e.g. le ciel). Enfin, cette séquence possède également des personnages en mouvement compliquant encore davantage le mouvement global de la vidéo.

(a) Football

(b) Coastguard

(c) Flower

(d) Stefan

(e) Bus

(f) Tempête

(g) News

(h) Foreman

(i) Container

(j) Paris

(k) Silent

(l) Table Tennis

Figure 3.1: Les 12 vidéos du corpus étudié, chaque vidéo est constitue de 150 images (252x288 pixels).

Les résultats que nous donnons dans ce qui suit ont été obtenus à l'aide de l'algorithme Full Search avec une taille de bloc T=16 et une fenêtre de recherche ± 7 pixels. Toutes les séquences que nous avons étudiées(figure 3.1) sont composées chacune de 150 images (352x288). Dans un premier temps, nous avons calculé la fréquence des vecteurs de mouvement (FVM) de chaque séquence comme le montre les figures suivantes. Les figures ci-dessous représentent la fréquence des vecteurs de mouvement en fonction de leur amplitude sur la fenêtre de recherche [-7, +7]. La figure 3.2 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Football*.

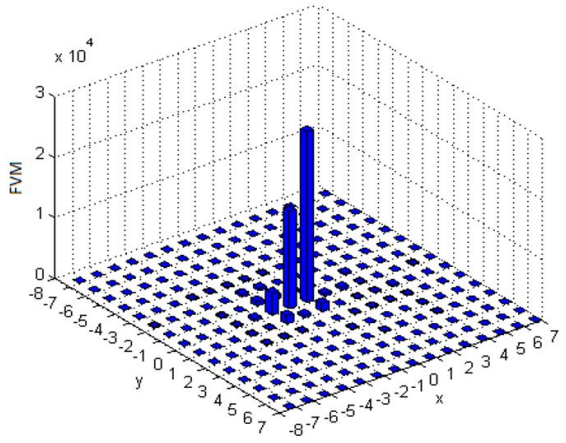


Figure 3.2: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Football*.

La FVM de la séquence *Football* montre que 27.02% des blocs ont un vecteur nul, 56.56% ont un vecteur sur l'axe des xx et 2.77% ont un mouvement sur l'axe des yy.
La figure 3.3 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Coastguard*.

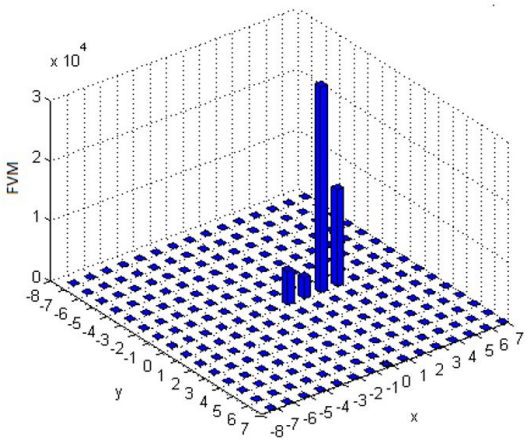


Figure 3.3: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Coastguard*.

On trouve à partir de la FVM de la séquence *Coastguard* que 9.34% des blocs ont un vecteur de mouvement nul, 90.4% ont un vecteur sur l'axe des xx et 0.25% ont un vecteur sur l'axe des yy. Nous déduisons que c'est un mouvement de translation selon xx.

La figure 3.4 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Flower*.

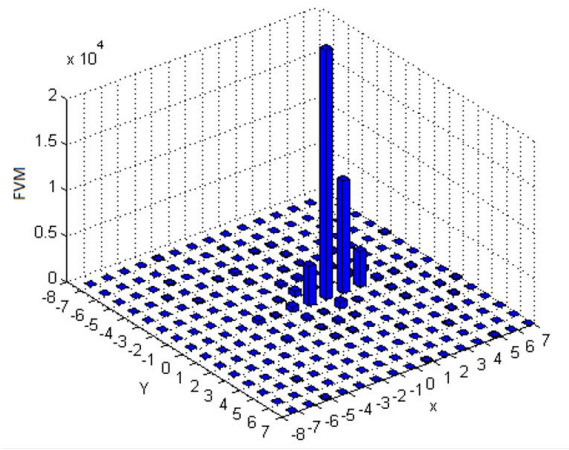


Figure 3.4: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Flower*.

On constate à partir de la FVM de la séquence Flower que 7.32% des blocs ont un vecteur de mouvement nul, 76.26% ont un vecteur sur l'axe des xx et 2.78% ont un vecteur de mouvement sur l'axe des yy.

La figure 3.5 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Paris*.

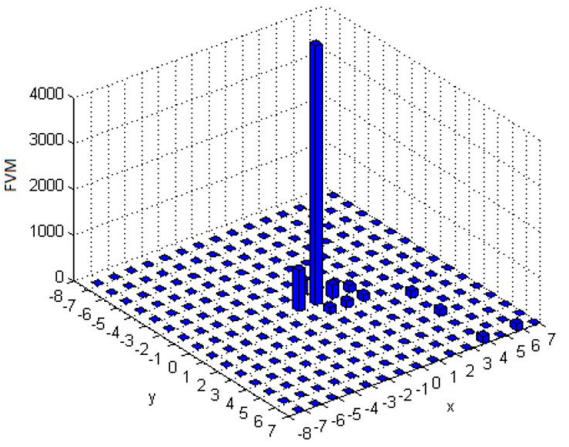


Figure 3.5: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Paris* (Pour une meilleure représentation graphique la fréquence FVM pour (0, 0) a été divisée par 10).

On distingue à partir de la FVM de la séquence *Paris* que 95.45% des blocs ont un vecteur

de mouvement nul, 2.27% ont un vecteur sur l'axe des xx et 0.76% ont un vecteur sur l'axe des yy. nous concluons que le mouvement est quasi – statique.

La figure 3.6 présente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Stefan*.

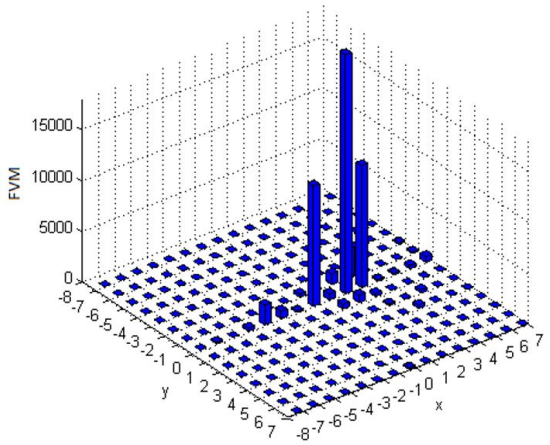


Figure 3.6: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Stefan*.

On voit à partir de la FVM de la séquence *Stefan* que 19.94% des blocs ont un vecteur de mouvement nul, 68.44% ont un vecteur sur l'axe des xx et 0.76% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est concentré sur l'axe des xx.

La figure 3.7 représente la FVM après estimation exhaustive dans la séquence *Bus*.

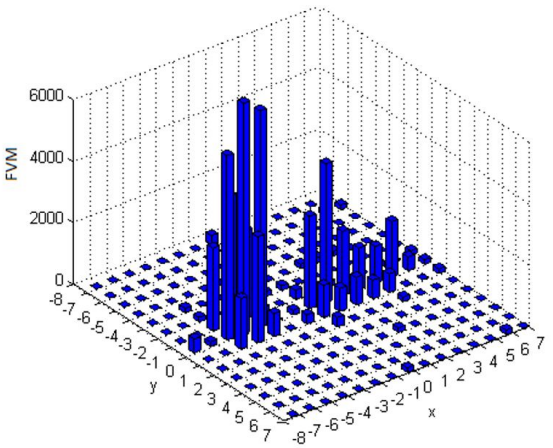


Figure 3.7: La FVM après estimation exhaustive sur la séquence *Bus*.

On observe à partir de la FVM de la séquence *Bus* que 5.05% des blocs ont un vecteur nul, 55.3% ont un vecteur sur l'axe des xx et 3.78% ont un vecteur de mouvement sur l'axe des yy.

Nous concluons que le mouvement se concentre sur et au voisinage de l'axe des xx.

La figure 3.8 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Tempête*.

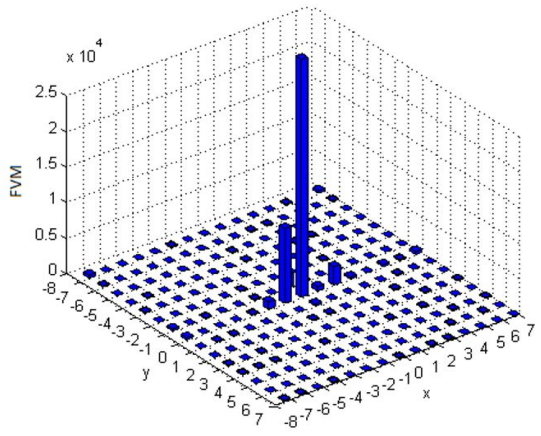


Figure 3.8: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Tempête* (Pour une meilleure représentation graphique la fréquence FVM pour (0, 0) a été divisée par 10).

On perçoit à partir de la FVM de la séquence *Tempête* que 56.31% des blocs ont un vecteur nul, 27.52% ont un vecteur sur l'axe des xx et 4.54% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est statique et translationnel sur l'axe des xx.

La figure 3.9 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Silent*.

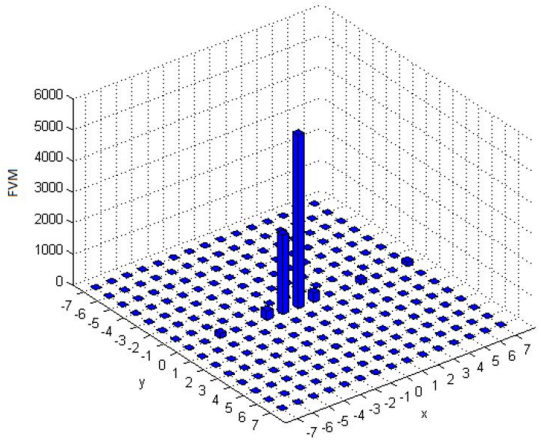


Figure 3.9: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Silent*.

On montre à partir de la FVM de la séquence *Silent* que 93.93% des blocs ont un vecteur de mouvement nul, 6.07% ont un vecteur sur l'axe des xx et 0% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est statique et translationnel sur l'axe des xx.

La figure 3.10 représente FVM obtenue à l'aide de l'algorithme FS dans la séquence *News*.

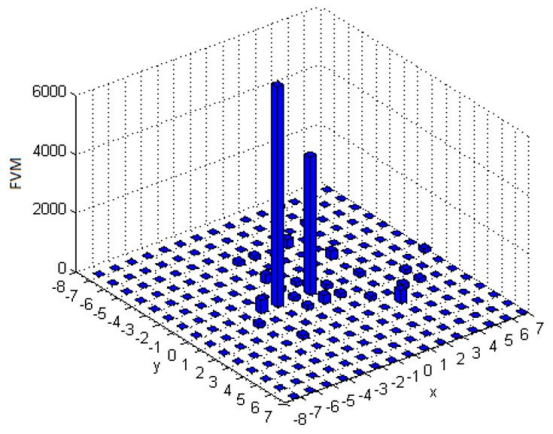


Figure 3.10: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *News* (Pour une meilleure représentation graphique la fréquence FVM pour (0, 0) a été divisée par 10).

On trouve à partir de la FVM de la séquence *News* que 79.29% des blocs ont un vecteur de mouvement nul, 14.39% ont un vecteur sur l'axe des xx et 1.01% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est statique et translationnel sur l'axe des xx.

La figure 3.11 représente FVM obtenue à l'aide de l'algorithme FS dans la séquence *Foreman*.

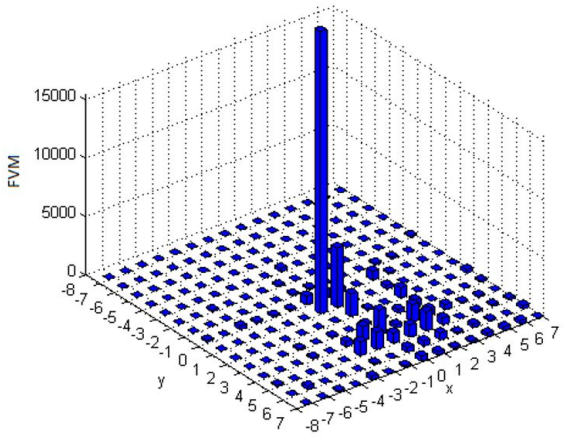


Figure 3.11: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Foreman*.

On constate à partir de la FVM de la séquence *Foreman* que 1.26% des blocs ont un vecteur de mouvement nul, 4.8% ont un vecteur sur l'axe des xx et 19.44% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est statique et sans une orientation bien définie.

La figure 3.12 représente la FVM obtenue à l'aide de l'algorithme FS dans la séquence *Container*.

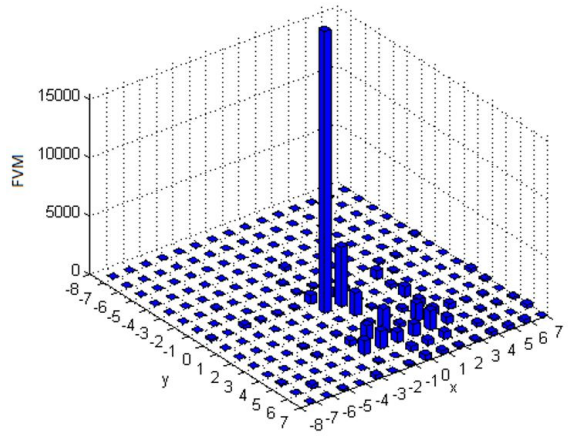


Figure 3.12: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Container* (Pour une meilleure représentation graphique la fréquence pour (0, 0) a été divisée par 10).

Pour la séquence *Container* 93.18% des blocs ont un mouvement nul, 5.3% ont un vecteur sur l'axe des xx et 0.25% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est statique et translationnel sur l'axe des xx.

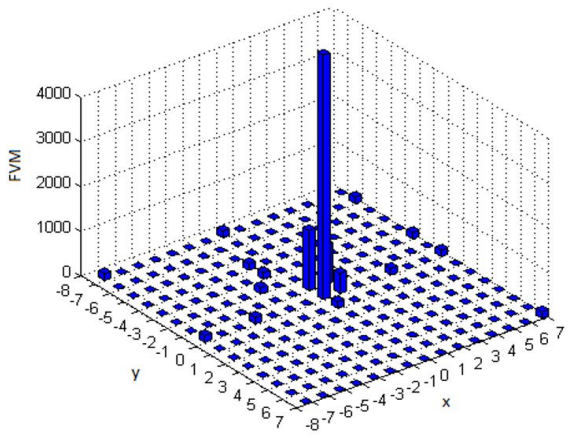


Figure 3.13: La FVM obtenue à l'aide de l'algorithme FS dans la séquence *Table tennis* (Pour une meilleure représentation graphique la fréquence pour (0, 0) a été divisée par 10).

La FVM de la séquence *Table tennis* (figure 3.13) 92.17% des blocs ont un vecteur de mouvement nul, 1.76% ont un vecteur sur l'axe des xx et 2.77% ont un vecteur sur l'axe des yy. Nous concluons que le mouvement est très faible ce qui correspond bien à la réalité.
La figure 3.14 représente la moyenne FVM obtenue à l'aide de l'algorithme FS dans les 12 séquences du corpus que nous avons utilisé.

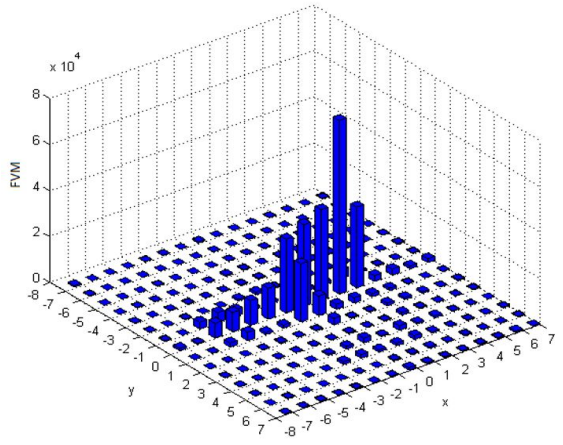


Figure 3.14: Allure moyenne de la FVM obtenue à l'aide de l'algorithme FS dans les 12 séquences du corpus.(Pour une meilleure représentation graphique la fréquence FVM pour (0, 0) a été divisée par 10).

L'étude statistique sur le corpus constitué de 12 vidéos donne les résultats suivants :

- 97.18% des vecteurs de mouvement se situent dans un disque de rayon 5 pixels,
- 92.15% à ±3pixels de centre.
- 50.06% des vecteurs de mouvement sont nuls. Nous avons remarqués qu'il y a une forte concentration de vecteurs de mouvement dans les axes (horizontalement et verticalement).
- Comme dans les travaux ultérieurs (S. Zhu and K. K. Ma, 2000), approximativement 80% des vecteurs de mouvement se situent dans un disque de rayon 2 pixels[59].
- On remarque aussi, comme (S. Zhu and K. K. Ma, 2000), la fréquence du vecteur de mouvement est relativement plus élevée sur les axes xx et yy (32.39% dans l'axe xx et 3.33% dans l'axe yy)[59].

Notre propos dans ce qui suit est de proposer, une méthode qui puisse améliorer les performances de Diamond Search en dehors du disque de rayon 2 pixels et qu'elle puisse exploiter le fait que la fréquence du vecteur de mouvement est élevée sur les axes xx et yy (ce qui correspond à des déplacements horizontaux ou verticaux respectivement).

Liste des tableaux 3.1: La fréquence des vecteurs de mouvement dans les images réelles.

Nb pixels Séquences	**0**	**±1**	**±2**	**±3**	**±4**	**±5**	**±6**	**±7**
Football	47.47	80.05	92.92	96.71	99.49	99.49	99.74	100
Coastguard	9.34	15.15	72.72	99.74	99.74	99.74	99.74	100
Flower	7.32	58.08	81.31	91.91	95.45	97.47	98.48	100
Stefan	19.94	22.47	66.41	95.45	96.71	97.47	98.23	100
Bus	5.05	17.17	23.73	45.45	74.74	95.95	97.22	100
Tempête	56.31	78.78	85.85	86.86	88.63	91.16	93.18	100
Silent	93.93	98.73	99.24	99.24	99.49	99.74	99.74	100
News	79.29	81.31	94.69	97.22	98.73	99.49	99.49	100
Paris	95.45	98.48	98.98	98.98	99.49	99.49	99.49	100
Foreman	1.26	52.77	56.81	63.13	75.50	88.63	93.43	100
Container	93.18	95.45	96.46	97.47	98.48	99.24	100	100
Table tennis	92.17	96.96	97.22	97.47	98.23	98.23	98.23	100

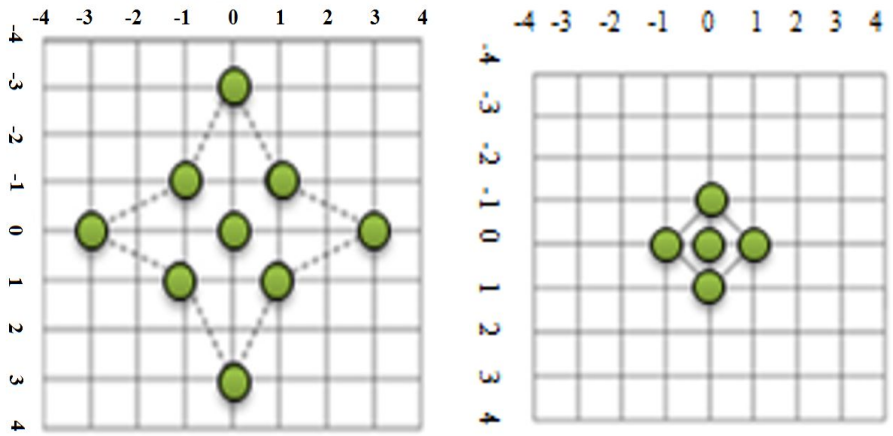
3.2.1 Analyse de la fréquence de distribution des vecteurs de mouvement

La fréquence de distribution des vecteurs de mouvement (VM) lorsque l'estimation se fait de manière exhaustive (tous les blocs de la fenêtre de recherche sont testés) a montré que les VM nuls sont les plus nombreux. Ils correspondent aux blocs qui ne bougent pas d'une image à l'autre. Les VM les plus fréquents sont aussi dans le voisinage du centre, on observe une forte concentration des VM sur les axes horizontaux et verticaux. Le tableau 3.1 donne la fréquence du vecteur de mouvement en fonction de son amplitude (nombre de pixels). On observe que pour les séquences à mouvement lent (*Football, Flower, Tempête, Silent, News, Paris, Container et Table tennis*) plus de 80% des vecteurs de mouvement ont une amplitude inférieure ou égale à 2 pixels. Cependant, ce n'est pas le cas pour les séquences à mouvement plus rapide *(Coastguard, Stefan, Bus et Foreman)* : elles nécessitent un disque de rayon supérieur à 2 pixels pour atteindre les 80%.

3.3 Proposition d'un nouvel algorithme Étoile-Diamant (ED).

La stratégie de recherche de l'algorithme que nous proposons *Étoile Diamant (ED)* est effectuée en deux phases (cf. figure 3.15). La première phase consiste à faire une recherche

grossière en *Étoile* (cf. figure 3.15 a) composée de neuf points. La deuxième recherche, cette fois-ci fine (cf. figure 3.15 b) est constituée de cinq points de recherche formant le *Petit Diamant (PD)*.



(a) *Etoile* (b) *Petit Diamant*

Figure 3.15: L'algorithme Étoile-Diamant (ED) est constitué par (a) une recherche *Étoile*, suivie par (b) une recherche *Petit Diamant (PD)*.

L'algorithme ED :

- **Phase I :** Le modèle *Étoile* (cf. figure 3.15 a) est centré sur l'origine du bloc courant et les neuf blocs sont testés (donc 9 SAD sont calculées), aller à phase II. On appellera par $b_{\min}$ le bloc correspondant à SADmin.
- **Phase II :** On utilise le modèle *PD* avec $b_{\min}$ étant le bloc central. Calculer SADmin correspondant au nouveau $b_{\min}$. Procéder ainsi jusqu'à obtenir $b_{\min}$ au centre de *PD*.

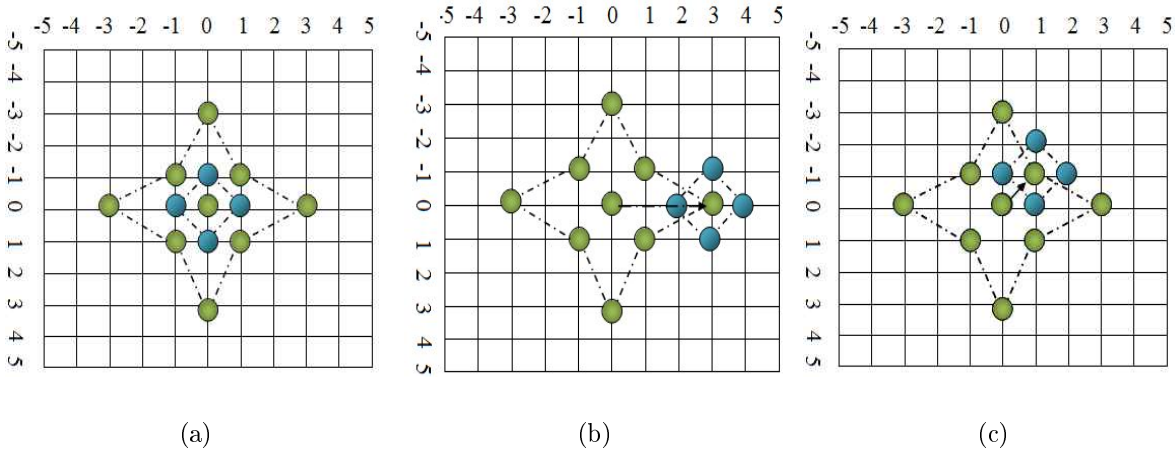


Figure 3.16: Les cas (a)(b) et (c) correspondent respectivement à SADmin au point central, horizontal et enfin sur la diagonale de *l'Étoile*.

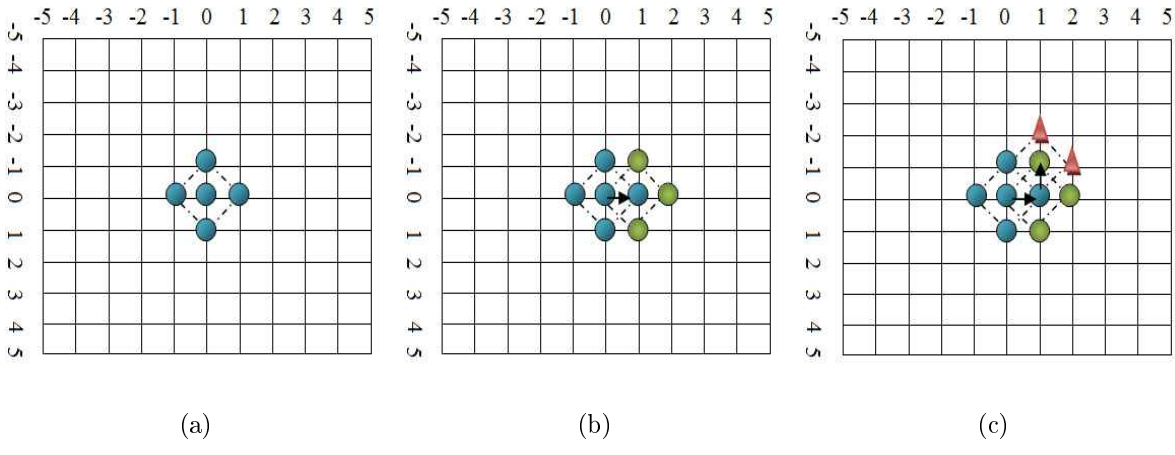


Figure 3.17: Chaque déplacement de *PD* (horizontal ou vertical) nécessite le calcul de 3 points, sauf si le déplacement antérieur était différent (voir figure 3.16 c) alors il nécessite 2 points à calculer.

Les figures 3.16 et 3.17 illustrent le fonctionnement de l'algorithme.

- La figure 3.16 considère les cas où l'optimum se trouve dans (0,0) en (a), (3,0) en (b) et enfin (1,1) en (c).
- La figure 3.16 (a) nous sommes en présence d'un mouvement nul. L'algorithme utilise le modèle *Étoile*. $b_{\min}$ est au centre de modèle *Étoile*. On utilise alors le modèle PD. $b_{\min}$ suivant est au centre de PD. Arrêt et fin de la procédure. VM(0,0) est 13 SAD ont été calculées.

– Figure 3.16 (b) nous sommes en présence d'un mouvement horizontal. Après l'utilisation du modèle *Étoile* on trouve que $b_{\min}$ correspond à (3,0). Après utilisation PD on trouve que le nouveau $b_{\min}$ correspond à (3,0) arrêt et fin. Cela à nécessite le calcule de 13 SAD.

– Figure 3.16 (c) dans ce cas le déplacement est sur la diagonale. Après *Étoile* $b_{\min}$ est à (1,1). Après *PD* $b_{\min}$ est au centre de *PD*, c'est le même. $b_{\min}$(1,1). Arrêt et fin. Le calcul nécessite le calcul de 13 SAD.

En conclusion , si le déplacement coïncide avec un point de l'*Étoile*, le calcul de 13 SAD est nécessaire. Dans le cas contraire, c'est à dire le déplacement est en dehors du modèle *Étoile*, il est nécessaire d'utiliser PD jusqu'à obtenir $b_{\min}$ (qui doit être le centre de PD).

La figure 3.17 étudie les 2 cas possibles.

– La figure 3.17 (b) montre qu'il faut calculer 3 SAD pour chaque déplacement horizontal.

– La figure 3.17 (c) montre qu'il faut, le cas d'un déplacement en diagonale, le calcul de deux SAD.

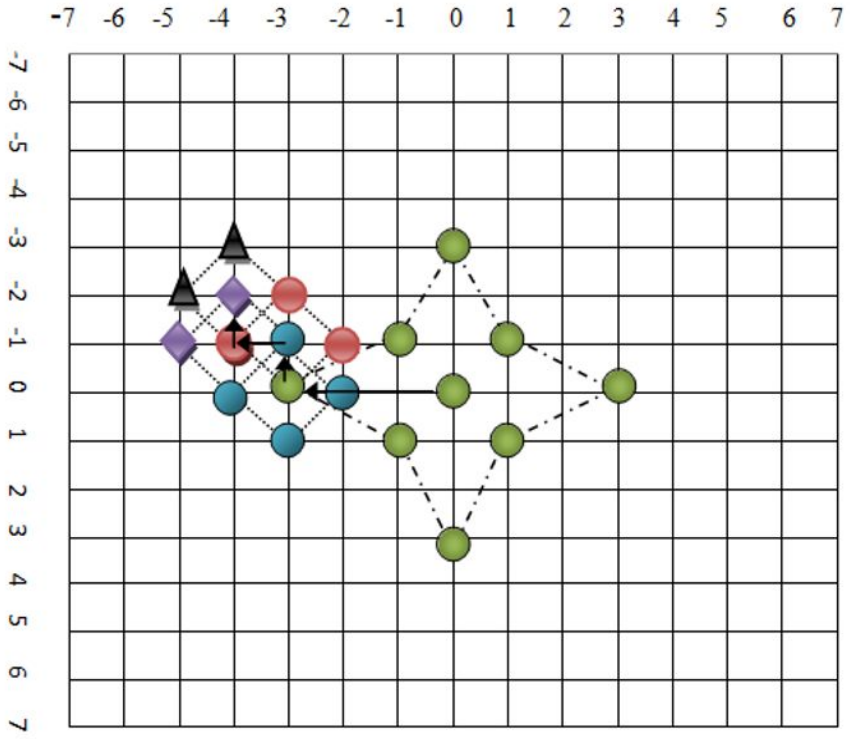
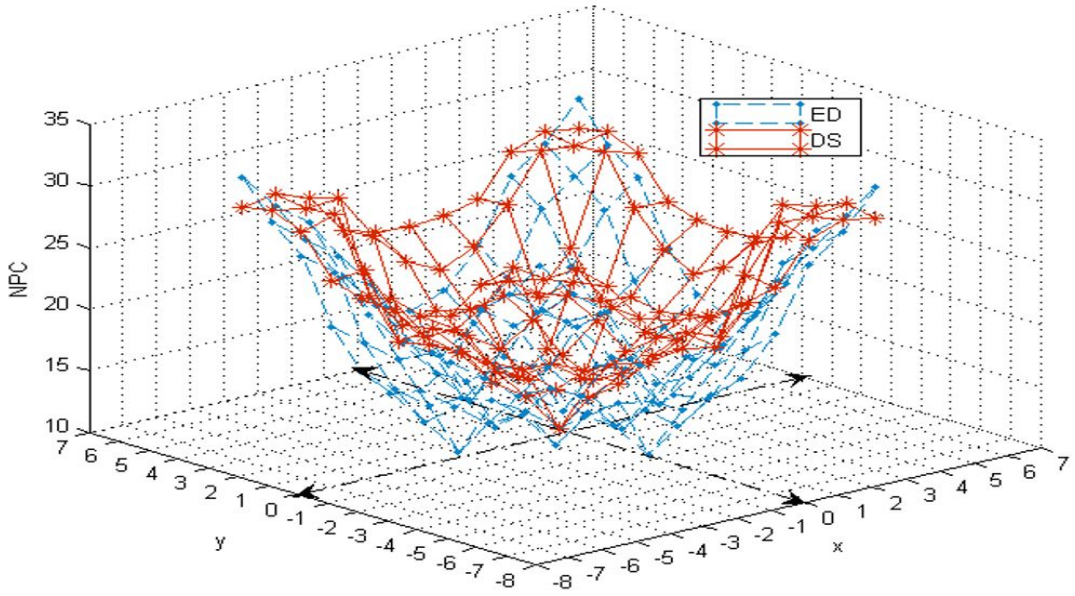


Figure 3.18: Exemple de solution se situant à (-4,-2) : ED effectue une recherche Étoile et 4 *PD* comme illustré dans la figure. Cela nécessite 9+4+3+2+2 = 20 points à calculer au lieu de 24 points (S. Zhu and K. K. Ma, 2000)[59].

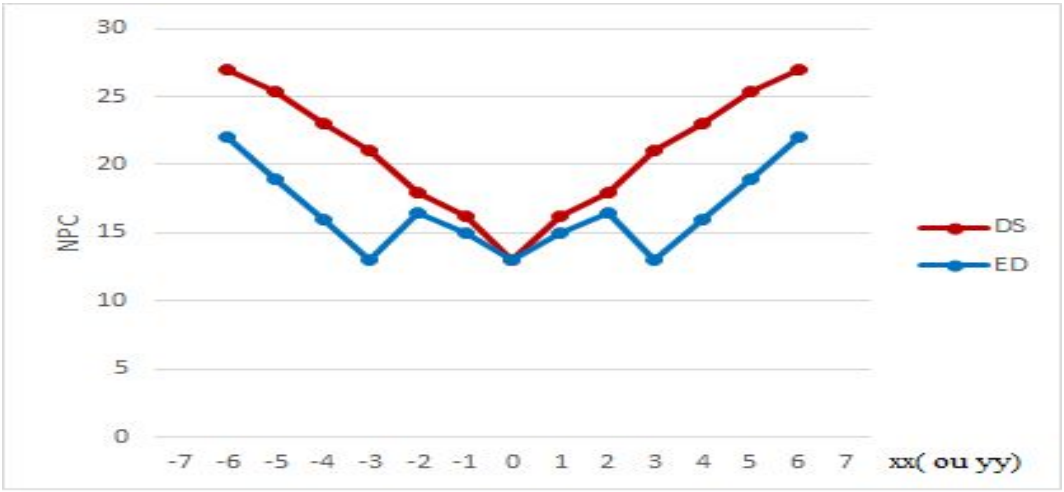
La figure 3.18 montre que le déplacement de *PD* nécessite à chaque fois le calcul de 2 à 3 points suivant que le déplacement est horizontal ou vertical respectivement. La figure 3.18, étudie le même exemple que celui donné dans (S. Zhu and K. K. Ma, 2000)[59]. Pour ED, un cheminement possible pour atteindre la solution optimale SADmin située à (-4,-2) est donné dans la figure 3.18. Le nombre de points calculés par l'algorithme proposé ED est de 9+4+3+2+2 =20 points au lieu de 24 points nécessité par DS (S. Zhu and K. K. Ma, 2000) [59].

Il est évident que plus nombre de points calculés pour la recherche du meilleur bloc est petit plus la complexité de l'algorithme est réduite. On montre que le nombre de points calculés (NPC) de ED et DS varie suivant l'amplitude et le sens du déplacement du vecteur de mouvement. La figure 3.19 a représente le nombre de points calculés par ED et DS pour les différents déplacements dans la fenêtre de recherche [-7, 7]. On remarque :

- Nombre de points calculés par ED est inférieur à celui de DS sur toute la fenêtre de recherche [-7, 7]. La différence augmente avec l'amplitude du vecteur de mouvement : en d'autres termes le gain en complexité obtenu par ED est d'autant plus grand que nous sommes en présence de séquences de mouvement rapide.
- la figure 3.19 b indique le nombre de points calculés par ED (en bleu) et le nombre de points calculés par DS (en rouge) sur l'axe des xx (ou yy), nous observons que le gain peut atteindre 30%.
- la figure 3.20 indique nombre de points calculés par ED (en bleu) et le nombre de points calculés par DS (en rouge) sur la diagonale. On remarque que ED pour les déplacements de deux et trois pixels sur la diagonale, il est de même performance que DS. Cependant pour les déplacements de 1 pixel, il est moins complexe. Notons que les déplacements diagonaux sont les moins fréquents (voir l'étude de la FVM effectuée sur un corpus de 12 séquences réelles). En conclusion, nous avons montré, dans cette section, que l'algorithme ED est moins complexe : il nécessite un nombre de points calculés plus petit (jusqu'à 30%) relativement à DS.



(a) NPC sur toute la fenêtre de recherche (-7, 7) calculé par les algorithmes ED et DS.



(b) NPC sur l'axe xx ou yy calculé par les algorithmes ED et DS.

Figure 3.19: NPC dans la fenêtre de recherche [-7, 7] en (a) et sur l'axe des xx (ou yy) en (b) calculé par les algorithmes ED et DS.

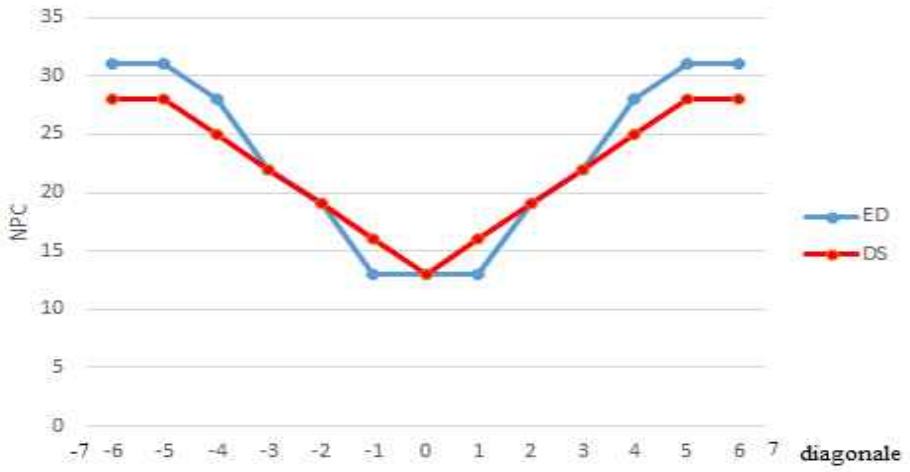


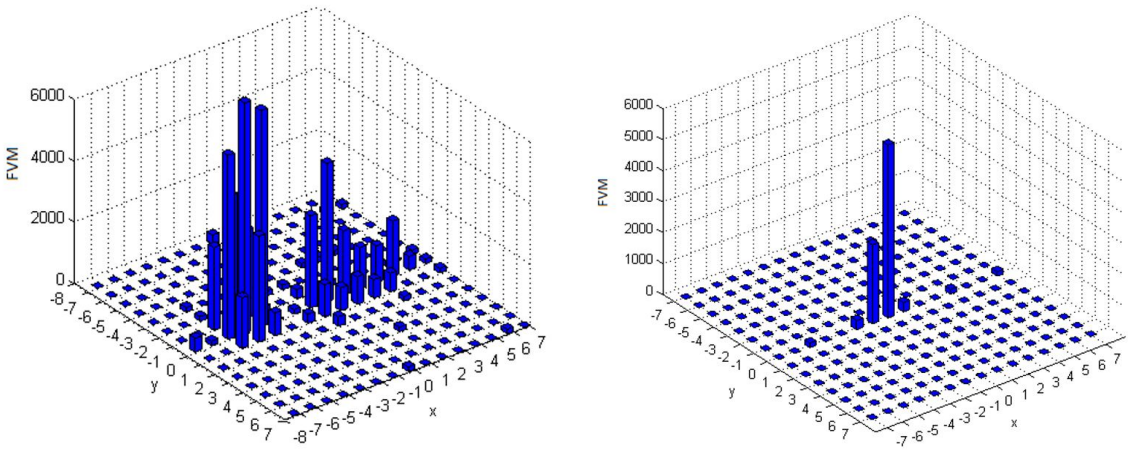
Figure 3.20: NPC sur la diagonale calculé par les algorithmes ED et DS .

3.3.0.1 Résultats et discussion

Il est intéressant de définir la complexité d'un algorithme par le nombre NPC_{moy} considéré comme la moyenne des NPC sur tous les blocs de la séquence traitée (chaque séquence étudiée étant constituée de 150 images). Il est clair que la complexité de l'algorithme est d'autant plus réduite que NPC_{moy} est faible. Nous avons effectué une étude comparative entre la méthode proposée Étoile-Diamant et les méthodes suivantes : FS [39] , NOCDS [62] , DS [59], 4SS [58], TSS [40]. Nous avons utilisé le PSNR comme paramètre de mesure sur la qualité et le NPC_{moy} pour la complexité de la méthode.

La Full Search est qualitativement meilleure : elle a le PSNR le plus élevé comparativement aux autres méthodes (voir Tableau 3.2). D'une manière générale, toutes les autres méthodes sont qualitativement équivalentes (NOCDS, DS, 4SS, TSS et Étoile-Diamant). Elles possèdent approximativement le même PSNR (cf. Tableau 3.2). Cependant, Étoile-Diamant possède un NPC_{moy} relativement plus petit de 7,86%, 19,36%, 24,14%, 40,66% et 93.23% que respectivement DS, NOCDS, 4SS, TSS et FS d'où une complexité moindre (cf. Tableau 3.3). Nous considérons comme séquence lente (par exemple *Silent*), une séquence dont 80% de ses vecteurs de mouvement sont à l'intérieur d'un disque de rayon 2 pixels, si ceux-ci ont en moyenne un rayon supérieur (par exemple *Bus*) elle est considérée comme séquence rapide c'est ce qui explique (cf. figure. 3.21 a) que la fréquence du vecteur de mouvement est plus étalée pour

la séquence *Bus* comparativement à celle de *Silent*. L'algorithme ED est plus performant que DS pour de grandes amplitudes du vecteur de mouvement (cf. figure 3.21 b.), et donc ED relativement à DS, est plus efficace pour les séquences rapides.



(a) Bus (b) Silent

Figure 3.21: La fréquence des vecteurs de mouvement pour les séquences *Bus* en (a) et *Silent* en (b) (la fréquence pour (0, 0) de *Silent* a été divisée par 10 dans la figure).

Liste des tableaux 3.2: Étude comparative à l'aide du PSNR sur le corpus des méthodes FS, TSS, 4SS, NOCDS, DS et ED.

Algorithmes *Séquences*	**FS**	**TSS**	**4SS**	**NOCDS**	**DS**	**ED**
Football	23,36	22,92	22,91	22,94	22,91	22.98
Coastguard	29,54	29,35	28,29	29,30	29,41	29,39
Flower	25,80	25,33	25,57	25,70	25,77	25,71
Paris	30,29	30,10	30,09	30,12	30,14	30,08
Stefan	24,97	24,61	24,13	23,93	24.10	24,54
Bus	24,44	22,88	21,91	21,82	21,97	22,93
Tempête	26,48	26,39	26,31	26,31	26,31	26,28
Silent	34.91	34,72	34,64	34,61	34,59	34,71
News	35,57	35,36	35,30	35,33	35,34	35,28
Foreman	32,89	32,39	32,52	32,40	32,67	32,46
Container	38,15	38,15	38.15	38.14	38,15	38,14
Table tennis	30,88	30,20	30,39	30,41	30,37	30,55
Moyen total	29,77	29,36	28.36	28.44	29,31	29,42

Liste des tableaux 3.3: Étude comparative à l'aide du NPC_{moy} des méthodes FS, TSS, 4SS, NOCDS, DS et ED sur le corpus étudié.

Algorithmes Séquences	**FS**	**TSS**	**4SS**	**NOCDS**	**DS**	**ED**
Football	204.28	23.31	18.22	19.35	16.14	14.93
Coastguard	204.28	23.33	18.57	18.96	16.70	14.18
Flower	204.28	23.31	18.39	20.26	15.90	13.94
Paris	204.28	23.21	16.20	13.36	12.84	12.56
Stefan	204.28	23.31	18.68	20.05	17.05	14.97
Bus	204.28	23.56	20.49	25.14	19.72	16.79
Tempête	204.28	23.27	16.32	13.68	13.11	13.03
Silent	204.28	23.21	16.37	13.73	13.14	12.90
News	204.28	23.21	16.19	13.37	12.83	12.54
Foreman	204.28	23.29	18.13	19.38	15.76	14.05
Container	204.28	23.22	15.93	12.30	12.40	12.32
Table tennis	204.28	23.30	17.33	16.12	14.47	13.63
Moyen total	204.28	23.29	17.56	17.14	15.00	13.82

Les figures 3.22 et 3.23 donnent une idée de l'évolution de la complexité des différents algorithmes suivant qu'on soit en présence d'une séquence d'images à mouvement rapide *Bus* et lente Silent.

Pour le mouvement rapide *Bus* figure 3.22, il est prévisible que ED donne les meilleurs résultats en terme de complexité par rapport aux autres algorithmes (NPC_{moy} pour ED est de 16.79 au lieu de 19.72 pour DS).

Pour le mouvement lent *Silent* figure 3.23 (NPC_{moy} pour ED est de 12.90 au lieu de 13.14 pour DS.

En conclusion, la réduction de la complexité obtenue par ED est d'autant plus grande que nous sommes en présence d'image rapide (image dont la distribution du vecteur mouvement est étalée avec des rayon > 2 pixels).

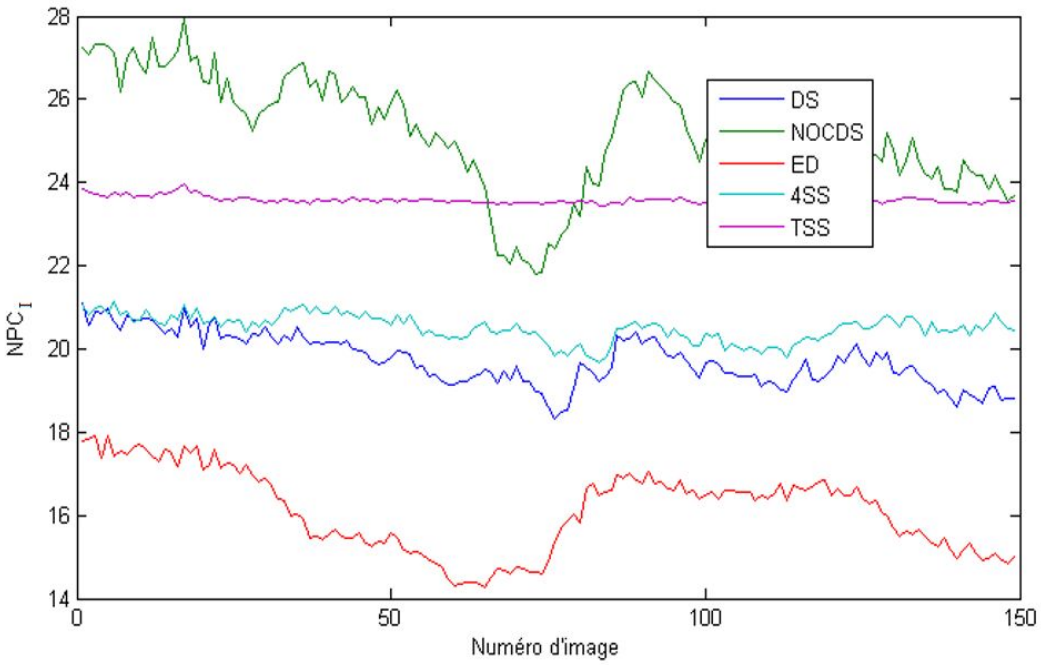


Figure 3.22: NPC_I des images de la séquence *Bus* (rapide).

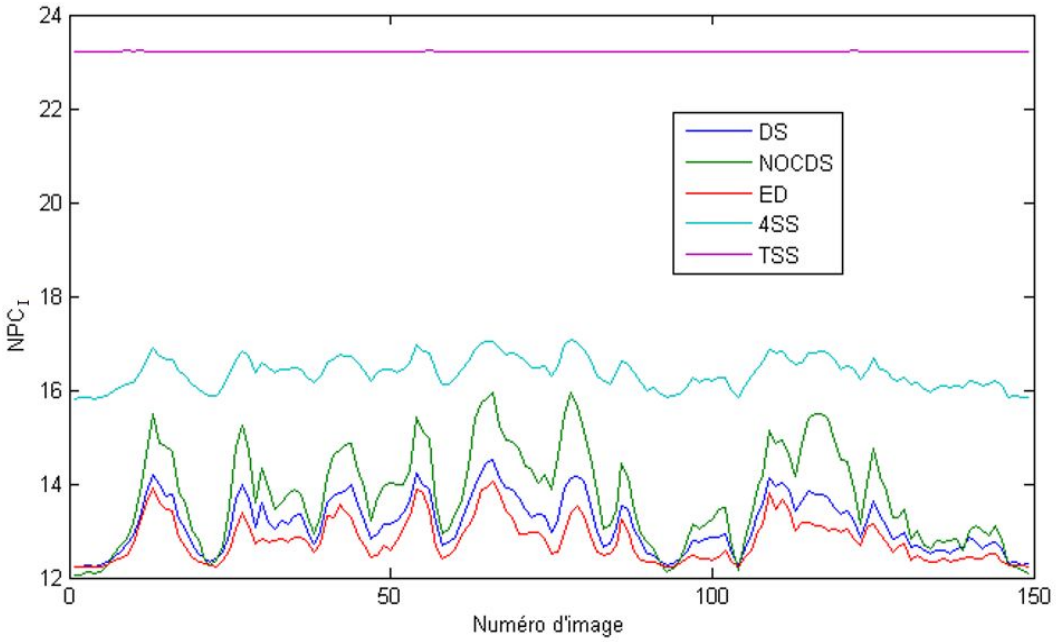


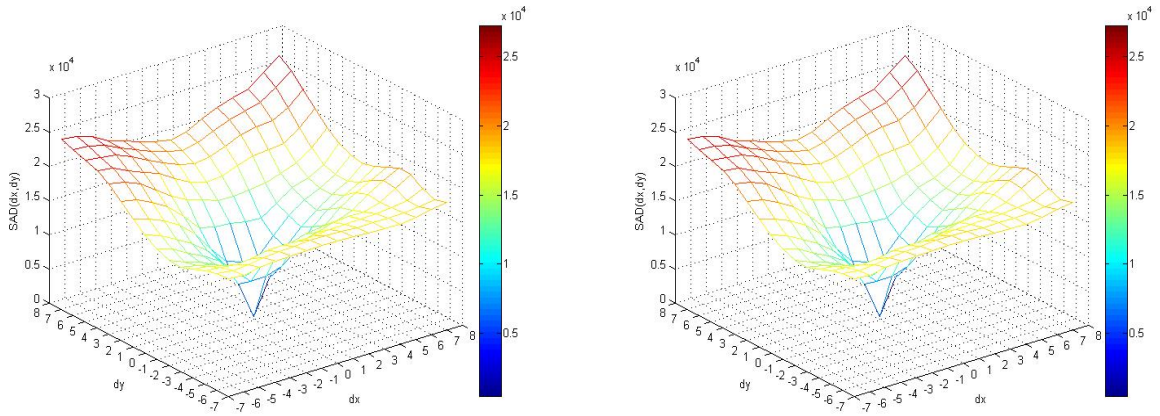
Figure 3.23: NPC_I des images de la séquence *Silent* (lente).

3.4 Proposition d'un nouvel algorithme en utilisant un seuil adaptatif (ED_S)

L'étude des différents algorithmes[40]−[41] nous a permis de faire un ensemble d'observations. L'erreur optimale Eopt, du meilleur bloc, est relativement petite lorsque le bloc n'est pas en déplacement. Eopt est d'autant plus élevée que le mouvement est élevé. Une étude a été effectuée montrant que cette observation est quasiment toujours versifiée. L'algorithme que nous allons présenter exploite cette idée.

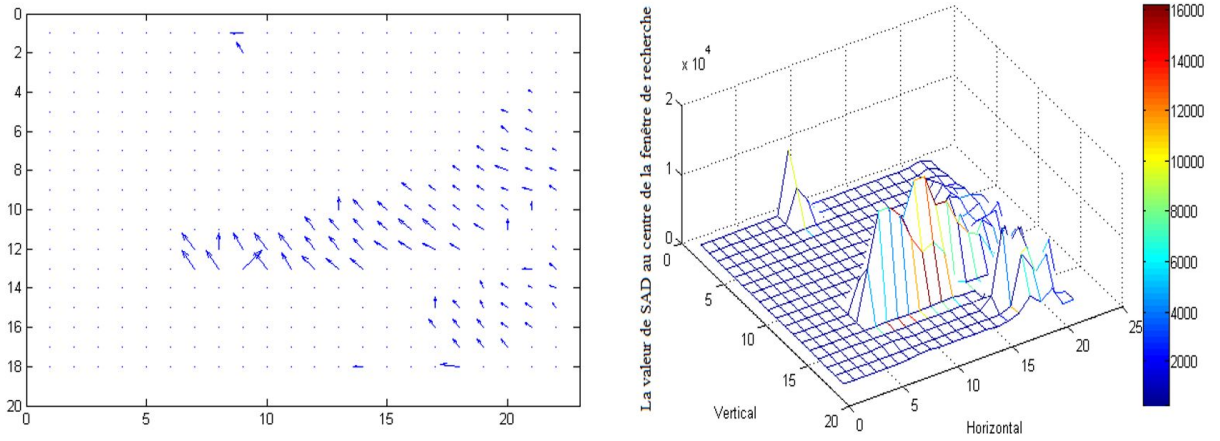
3.4.0.2 Méthode de détermination de seuil

On observe que la fréquence des vecteurs de mouvement FVM dans les images réelles est élevée pour les vecteurs de mouvement nuls. Lors de la recherche de meilleure erreur dans la fenêtre de recherche, l'erreur d'appariement diminue pendant lorsqu'on se déplace vers le minimum global (figure 3.24).



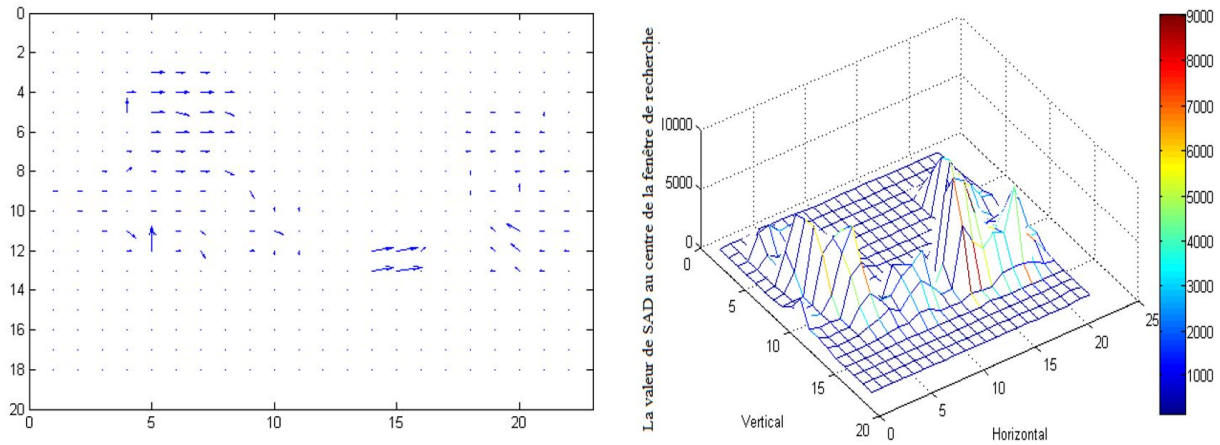
(a) *News* (b) *Flower VM* $\neq$ *0*

Figure 3.24: (a) SAD (dx,dy) erreur de mise en correspondance pour un bloc statique et (b) dynamique à partir d'une paire d'images de la séquence *News* et *Flower* (bloc 16x16 et une fenêtre de recherche de taille ± 7 pixels). L'erreur minimale pour *News* (443) est nettement inférieure à celle pour *Flower* (2556).



(a) (b)

Figure 3.25: Champs de vecteurs de mouvement et erreurs de mise en correspondance au centre de la fenêtre de recherche pour les blocs statiques et dynamiques obtenus à partir de la même paire d'images de la séquence *Table tennis*, (SAD bloc 16x16 et une fenêtre de recherche ± 7 pixels). On remarque il y a une forte corrélation les figures (a) et (b). Les blocs statiques en (a) correspond à une erreur négligeable, en (b) ce qui n'est pas le cas pour les blocs en mouvement.



(a) (b)

Figure 3.26: Champs de vecteurs de mouvement et erreurs de mise en correspondance au centre de la fenêtre de recherche pour les blocs statiques et dynamiques obtenus à partir de la même paire d'images de la séquence *Paris* ,(SAD bloc 16x16 et une fenêtre de recherche de taille ± 7 pixels). Même observation pour la figure 3.25(b).

Nous avons effectué une étude, sur plusieurs dizaines d'exemples, la SAD au centre de la fenêtre de recherche. Pour les blocs statiques est négligeable alors que celle des blocs dynamiques est importante voir les figures 3.25, et 3.26). Par conséquent, une valeur seuil S peut être

employée. Si on peut du fait que la valeur de l'erreur centrale est un paramètre permettant de décider s'il y a (ou non) un déplacement alors on fera comme suit : Si l'erreur centrale $\leq S$ on considéra le bloc est statique. Aucun recherche ne se fera (aucune SAD n'est calculée). Si l'erreur central $>$ S on fait une recherche de l'erreur minimale.

Pour chaque séquence la valeur de l'erreur de mise en correspondance blocs varie cela implique un seuil différent selon le contenu de mouvement de la scène. L'utilisation d'un seuil approprié est nécessaire. Pour une séquence d'images donnée le seuil est calculé de la manière suivante :

A étant la matrice de toutes les valeurs de $SAD(0,0)$ au centre de chaque fenêtre de recherche entre les deux premières images de la même séquence.

$$A = \begin{bmatrix} SAD_{1,1}(0,0) & \cdots & SAD_{1,n}(0,0) \\ \vdots & \ddots & \vdots \\ SAD_{m,1}(0,0) & \cdots & SAD_{m,n}(0,0) \end{bmatrix} \tag{3.1}$$

Avec **m, n** correspondent à la longueur de la ligne et de la colonne de A.

$$S = Max\,(Min\,(A)) \tag{3.2}$$

$$(SM, AM) = \left\{ \begin{array}{l} SM,\ if\ SAD\,(0,0) \leq S \\ AM,\ if\ SAD\,(0,0) > S \end{array} \right\} \tag{3.3}$$

Avec : *SM :* Sans Mouvement.

AM : Avec Mouvement.

Remarque ; ce seuil S est déterminé une seul fois pour chaque séquence et considéré valable pour toutes la séquence (pour chaque séquence on a un seuil différent).

Les principales phases de l'algorithme ED$_S$:

- Si SAD (0,0) $\leq S$,

  Alors le bloc est statique,

- Si non, procéder à la recherche du vecteur mouvement à l'aide de l'algorithme ED.

3.4.0.3 Résultats et discussion

Un ensemble de résultats obtenus sont représentés dans le tableau 3.4. Le paramètre NPC_{moy} permet d'évaluer la complexité de l'algorithme sur toute une séquence (150 image). On observe (Tab 3.4) que la FS est la plus complexe et le nombre d'opération effectué est constant pour les 12 séquences. On remarque aussi que les moins couteuse en termes de complexité sont ED et ED_S. Il est important de signaler que pour la séquence News, l'algorithme ED_S a permis de diviser le nombre d'opération de plus de 100 (NPC_{moy}=204.28 pour FS contre 1.98 pour ED_S). D'une manière générale, les algorithmes ED et ED_S présentent une complexité plus réduite comparativement aux algorithmes FS, TSS [40], SES [57], 4SS [58], NOCDS [62], HEX [41], DS[59] et de cela , comme l'indique le tableau 3.5, pour une qualité de l'image pratiquement identique.

Le tableau 3.5 présente une évaluation de la qualité des séquences obtenue à l'aide du PSNR, du PSNR-HVSM [37] et de SSIM [38]. On observe que FS présente des erreurs les plus intéressantes relativement aux autres méthodes. Cependant, les différences obtenues ne sont pas semblables à l'œil nu : on peut dire que les images obtenues pour la plupart des méthodes sont pratiquement de "même qualité".

Liste des tableaux 3.4: NPC_{moy} :Étude comparative des algorithmes FS, TSS, SES, 4SS, NOCDS, HEX, DS, ED et ED_S.

Algorithmes *Séquences*	**FS**	**TSS**	**SES**	**4SS**	**NOCDS**	**HEX**	**DS**	**ED**	**ED_S**
Football	204.28	23.31	16.39	18.22	19.35	16.73	16.14	14.93	11.72
Coastguard	204.28	23.33	16.37	18.57	18.96	16.97	16.70	14.18	14.09
Flower	204.28	23.31	16.09	18.39	20.26	16.99	15.90	13.94	9.68
Paris	204.28	23.21	16.94	16.20	13.36	11.51	12.83	12.56	5.05
Stefan	204.28	23.31	16.40	18.68	20.05	17.50	17.05	14.97	11.72
Bus	204.28	23.56	16.28	20.49	25.14	21.81	19.72	16.79	12.36
Tempête	204.28	23.27	16.93	16.32	13.68	12.05	13.11	13.03	4.52
Silent	204.28	23.21	16.89	16.37	13.73	11.82	13.14	12.90	3.85
News	204.28	23.21	16.96	16.19	13.37	11.35	12.83	12.54	1.98
Foreman	204.28	23.29	16.28	18.13	19.38	16.29	15.76	14.05	11.50
Container	204.28	23.22	17.05	15.93	12.30	10.63	12.39	12.32	2.21
Table tennis	204.28	23.30	16.60	17.33	16.12	13.99	14.47	13.63	9.14

Liste des tableaux 3.5: Étude comparative de TSS, SES, 4SS, NOCDS, HEX, DS et ED_S à l'aide du PSNR, PSNR_HVSM et SSIM.

Algorithmes *Séquences*	Critères d'évaluations	**FS**	**TSS**	**SES**	**4SS**	**NOCDS**	**HEX**	**DS**	**ED_S**
Football	PSNR	23,35	22,92	22,31	22,91	22,94	22,89	22,90	22,87
	PSNR_HVSM	19,98	19,49	18,74	19,41	19,43	19,35	19,36	19,35
	SSIM	0,8050	0,7896	0,7946	0,7959	0,7974	0,7958	0,7969	0,7946
Coastguard	PSNR	29,54	29,35	29,17	28,29	29,30	29,38	29,40	29,44
	PSNR_HVSM	28,56	28,29	28,06	28,31	28,20	28,33	28,38	28,44
	SSIM	0,8961	0,8893	0,8827	0,8900	0,8898	0,8907	0,8912	0,8921
Flower	PSNR	25,80	25,33	25,26	25,57	25,70	25,73	25,76	25,76
	PSNR_HVSM	25,63	30,95	30,80	25,37	25,50	25,54	25,58	25,58
	SSIM	0,9363	0,9201	0,9172	0,9269	0,9329	0,9335	0,9350	0,9327
Paris	PSNR	30,28	30,10	29,85	30,09	30,12	30,10	30,13	30,14
	PSNR_HVSM	28,42	28,18	27,73	28,1	28,10	28,05	28,11	28,12
	SSIM	0,9588	0,9579	0,9562	0,9584	0,9586	0,9588	0,9587	0,9588
Stefan	PSNR	30,96	30,61	23,95	30,13	23,93	30,08	30,10	30,38
	PSNR_HVSM	22,46	22.00	21,16	21,37	21,11	21,29	21,32	21,69
	SSIM	0,8693	0,8547	0,8356	0,8356	0,8346	0,8390	0,8397	0,8477
Bus	PSNR	30,44	22,88	21,66	21,91	21,82	21,89	21,96	22,22
	PSNR_HVSM	21,73	19,76	18,14	18,32	18,18	18,28	18,35	18,79
	SSIM	0,8630	0,8101	0,7437	0,7549	0,7510	0,748	0,7526	0,7512
Tempête	PSNR	26,47	26,39	26,28	26,31	26,31	26,31	26,30	26,30
	PSNR_VHSM	30,39	30,26	30,08	30,12	30,11	30,10	30,09	30,02
	SSIM	0,8908	0,8910	0,8902	0,8913	0,8917	0,8918	0,8918	0,8892
Silent	PSNR	34,91	34,72	34,32	34,64	34,61	34,58	34,59	34,59
	PSNR_VHSM	31,84	31,61	31,07	31,49	31,43	31,38	31,39	31,41
	SSIM	0,9585	0,9574	0,9555	0,9578	0,9580	0,9579	0,9582	0,9581
News	PSNR	35,56	35,36	35,04	35,30	35,33	35,31	35,34	35,11
	PSNR_HVSM	33,13	32,86	32,44	32,76	32,81	32,78	32,81	32,51
	SSIM	0,9737	0,9735	0,9725	0,9736	0,9736	0,9736	0,9737	0,9696
Foreman	PSNR	32,89	32,39	31,98	32,52	32,40	32,25	32,66	32,55
	PSNR_HVSM	31,62	30,92	30,30	31,11	30,98	30,73	31,30	31,13
	SSIM	0,9307	0,9147	0,9100	0,9189	0,9175	0,9152	0,9219	0,9205
Container	PSNR	38,15	38,15	38,14	38,14	38,14	38,14	38,14	38,14
	PSNR_VHSM	40,48	40,47	40,47	40,47	40,47	40,47	40,47	40,47
	SSIM	0,9753	0,9752	0,9752	0,9752	0,9752	0,9752	0,9752	0,9752
Table tennis	PSNR	30,87	30,20	29,60	30,39	30,41	30,30	30,37	30,45
	PSNR_HVSM	28,30	27,45	26,67	27,57	27,57	27,42	27,48	27,64
	SSIM	0,9147	0,8953	0,8841	0,9115	0,9130	0,9114	0,9123	0,9127

3.5 Conclusion

Dans ce chapitre une étude statistique de la distribution des vecteurs de mouvement a été effectuée sur un corpus contient 12 différents types de séquences d'images. De l'analyse de la FVM nous avons proposé deux nouvelles méthodes d'estimation de mouvement. La première méthode ED procède en deux étapes : la première utilise une forme d'*Étoile*, effectue une recherche grossière, la seconde utilise une forme de *Petit Diamant PD* affine le résultat de la recherche. La notion de nombre de points calculés NPC est défini pour la première fois dans le présent document. Ce paramètre permet de mesurer la complexité de l'algorithme. Une étude comparative avec les méthodes les plus visées FS, TSS, 4SS, NOCDS et en particulier la méthode de Diamant a montré que l'algorithme ED est moins complexe que Diamant : il y a une amélioration de 7,86% en moyenne sur l'ensemble du corpus utilisé dans le présent document (12 séquences de 150 images). Il a également été montré que des séquences rapides *(Bus)* l'amélioration en pourcentage peut atteindre 14,85%. De plus une estimation de mouvement rapide en utilisant un seuil adaptatif est défini à partir de l'erreur de mise en correspondance pour éliminer les blocs statiques du processus de l'estimation de mouvement est présentée dans ce chapitre, ce qui diminue remarquablement le nombre des points calculés. Par conséquent, notre deuxième méthode rend le procès d'estimation de mouvement plus rapide et réduit la complexité de calcul. L'étude comparative montre que notre deuxième méthode ED_S réduit d'environ (30-70%), le nombre de points calculés par bloc avec une même qualité de l'image par rapport aux autres algorithmes. L'algorithme ED_S est un bon candidat pour les applications de codage vidéo en temps réel.

CHAPITRE 4

DÉTECTION DES OBJETS MOBILES PAR LES MÉTHODES D'ESTIMATION DE MOUVEMENT

4.1 Introduction

La vidéo-surveillance intelligente (smart video surveillance) consiste à identifier automatiquement des objets, des comportements ou des événements particuliers dans des séquences vidéo. Elle analyse et transforme les données issues d'une (ou plusieurs) caméra en une interprétation sémantique directement exploitable par un opérateur humain. Par exemple, lorsqu'une anomalie est détectée, le système doit envoyer une alerte au personnel afin qu'il puisse prendre une décision adéquate. Cependant, le système de la vidéo surveillance qui inclut l'enregistreur vidéo nécessite un temps de calcul élevé et une grande capacité mémoire. Dans ce chapitre nous proposons une approche de détection des objets en mouvement pour optimiser la compression vidéo lors de la surveillance en se basant sur le champ de vecteurs de mouvement estimé par une méthode de mise en correspondance de blocs utilisé dans la compression vidéo [59].

La section 4.2 est consacrée à la détection des objets en mouvement dans une séquence vidéo.

La Section 4.3 présente un bref état de l'art de l'ensemble de méthodes de détection d'objet en mouvement que nous classons en trois types : les méthodes basées contour, les méthodes de flot optique et les méthodes basées sur la distribution des pixels [65] - [66]. Les sections 4.4 et 4.5 sont consacrées à la méthode proposée ainsi que son implémentation. Finalement, une étude comparative montre l'intérêt de notre propos.

4.2 Détection d'objets en mouvement

La plupart des méthodes de détection se basent sur l'une des approches suivantes :

– **Modélisation de l'arrière-plan (Background Modling)**

Cette catégorie regroupe toutes les méthodes de détection de mouvement qui consistent à créer un modèle de l'arrière-plan de la scène filmée (sans aucun objet mobile). Ce modèle peut être une image crée à partir des pixels observés à différents instants de la séquence vidéo [67].

– **Soustraction de l'arrière-plan (Background subtraction)**

La soustraction de l'arrière-plan est l'opération qui suit logiquement la modélisation de l'arrière-plan afin d'obtenir une détection de mouvement. Si le modèle de l'arrière-plan est une image, une différence en valeur absolue entre ce modèle et l'image courante est effectuée afin d'obtenir une détection de mouvement.

– **Segmentation de (par le) mouvement (Motion[-based] segmenta-tion)**

Cette tâche va au-delà de la détection de mouvement puisqu'il s'agit de segmenter chaque image en régions qui présentent une homogénéité du mouvement apparent. Cette opération est généralement réalisée à partir d'une estimation du flot optique [68] ou des dérivées spatio-temporelles de l'intensité lumineuse [69].

4.3 État de l'art

Dans la dernière décennie, de nombreuse méthodes de la détection d'objet en mouvement ont été proposées [65] - [66]. Nous classons ces méthodes en trois types : les méthodes basées contour, les méthodes de flot optique et les méthodes basées sur la distribution des pixels.
Les approches basées contour peuvent efficacement détecter les objets en mouvement de diffé-

rentes tailles et de formes, et semblent être insensibles aux variations d'illumination [65]. Parmi ces méthodes nous citerons les méthodes de level set [70], les contours actifs [71] et contours actifs géodésiques [72]. Cependant, ces méthodes sont coûteuses en termes de calcul.

Les approches fondées sur le flot optique d'objet en mouvement consistent à estimer le vecteur de mouvement de chaque pixel dans la séquence vidéo. Elles représentent l'image comme un espace de champ du vecteur de mouvement. Les méthodes présentées dans[73], [74] permettent la détection de mouvement à l'aide du flot optique. Ces méthodes peuvent détecter avec précision le mouvement (gradient d'intensité), mais le mouvement qui est tangent au gradient d'intensité ne peut pas être bien représenté. En outre, les méthodes basées flot optique souffrent également du problème d'éclairage, de sensibilité au bruit et coût de calcul élevé. Elles ne s'apprêtent pas aux applications temps réel.

Les approches basées distribution gaussien (ou la soustraction d'arrière-plan), consistent à estimer une trame d'arrière-plan de la scène en se basant sur la distribution de pixels. L'objet dans la trame courante peut être détecté par la soustraction de la trame courante avec la trame de fond. L'article [75] propose une méthode paramétrique qui consiste à construire un modèle de l'arrière-plan en tant que mélange de gaussiennes (Mixture of Gaussien (MoG)).

Dans le même contexte, d'autres approches similaires ont été proposées dans [76], [77], [78] et [79]. Cependant, l'estimation de modèle de l'arrière plan nécessite un temps de calcul élevé ainsi qu' une grande sensibilité à la variation de la lumière et au bruit. Lorsque les images de fond sont complexes, ces approches paramétriques peuvent échouer.

Dans l'article [80] on trouve un modèle non paramétrique de l'arrière-plan, l'avantage de cette méthode est qu'elle permet une meilleure précision pour un même taux de calcul, un autre avantage de ce type d'approche est l'intégration des contraintes spatiales dans la construction de la classification du premier plan. Plusieurs autres algorithmes de construction de modèle d'arrière-plan non paramétrique sont présentés dans [81], [82] et [66]. Il s'avère que les méthodes paramétriques sont plus efficaces que les méthodes non paramétriques [80]-[66]. On s'est surtout intéressé à la méthode paramétrique MOG (Mixture of Gaussien) qui nous semble la plus efficace comparativement aux autres méthodes. Nous utilisons ses résultats pour l'étude comparative que nous donnons dans la suite de ce travail. Nous proposons dans ce chapitre

une nouvelle approche efficace pour la Détection d'Objet Mobile avec CHamps de Mouvement (DOM-CHM) en utilisant un algorithme de mise en correspondance de blocs (Block Matching Algorithm)[59].

4.4 Proposition d'une nouvelle méthode de détection de mouvement : DOM-CHM.

Comme nous l'avons vu, la plupart des méthodes de détection de mouvement travaillent directement sur l'image et majoritairement, elles opèrent au niveau du pixel et rarement au niveau de bloc. Il s'avère que l'utilisation du bloc est moins coûteuse en temps de calcul. C'est ce qui explique que la méthode que nous proposons utilise une approche basée bloc. L'idée de base que nous exploitons est la suivante : le champ de mouvement différencie les blocs en mouvement de ceux qui sont statiques. Nous utilisons cette observation pour construire l'image de fond. Les étapes de l'algorithme proposé sont détaillées dans la section suivante.

4.4.1 Estimation de mouvement.

La première phase de notre algorithme DOM-CHM consiste à estimer les vecteurs de mouvement entre deux trames successives avec la méthode Diamant[59].

La figure 4.1 présente deux trames successives de la séquence *Sofa*. En (c) nous trouvons le champ de de mouvement. (d) présente l'image reconstruite de fond.

La caractérisation de l'objet en mouvement est obtenue par binarisation. A partir du champ de mouvement (c) on dira que l'objet en mouvement est défini par l'ensemble des vecteurs de mouvement non nuls. Il s'en suit que l'image de fond est définie par les vecteurs de mouvement nuls.

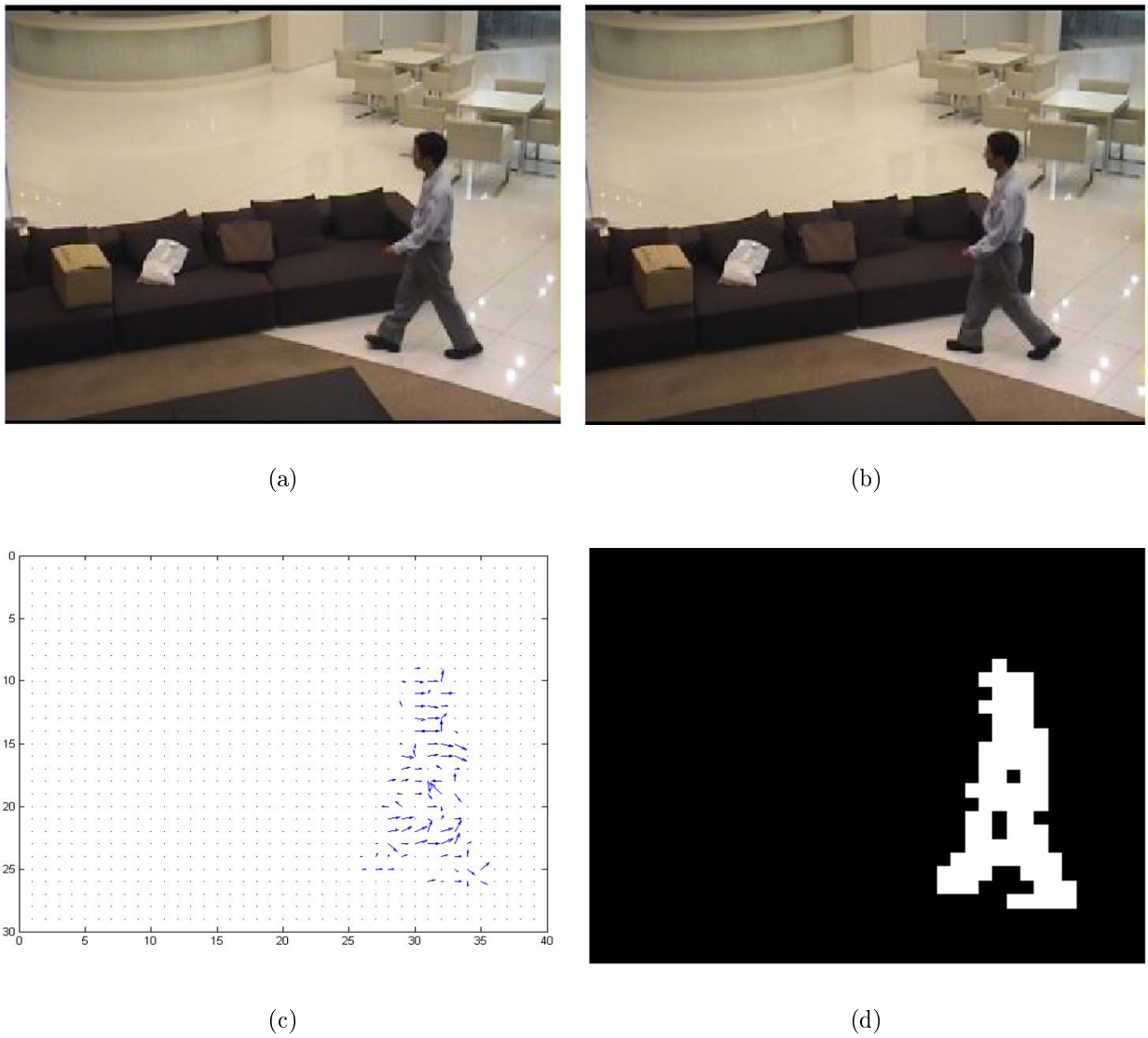


Figure 4.1: (a) et (b) représentent deux trames successives de la séquence *Sofa*, (c) Présente le champ de vecteurs de mouvement en utilisant l'algorithme Diamant avec seuillage (bloc de taille 8x8 pixels et la fenêtre de recherche ± 7 pixels). (d) l'image reconstruite après binarisation.

4.4.2 La reconstruction de l'image de fond (background) à partir de champ de vecteurs de mouvement.

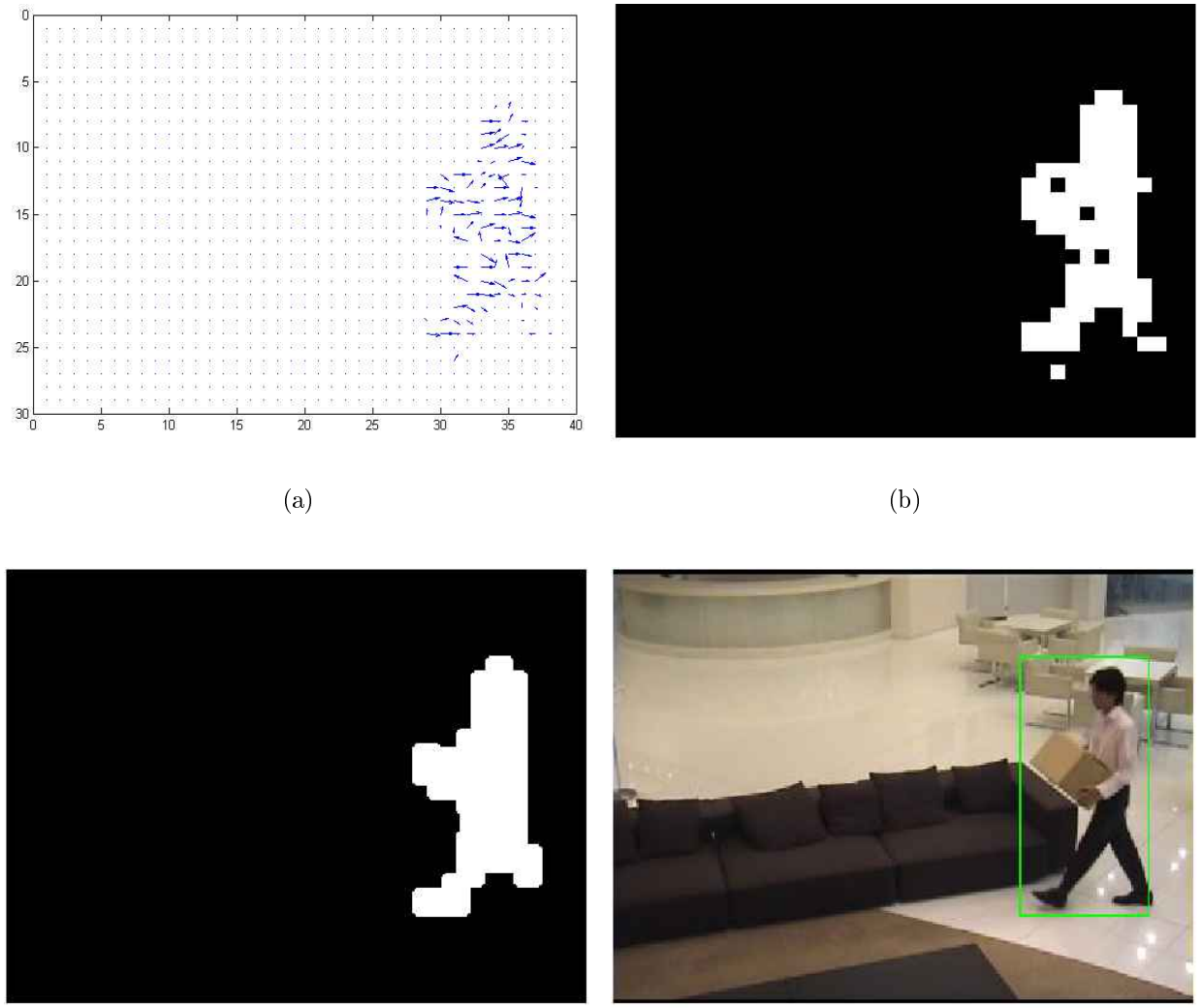
Lorsque la caméra vidéo est fixe, une des techniques les plus classiques exploite une modélisation de la partie fixe de la scène (appelé le fond ou "background") pour mettre en évidence les régions de l'image correspondant à des objets en mouvement dans la scène (appelé le premier plan ou "foreground"). La reconstruction de l'image fond dans notre propos se fait par des vecteurs de mouvement estimés entre deux images. C'est-à-dire que l'on reprend les blocs de

l'image précédente indiqués par les vecteurs de mouvement. Donc les blocs en mouvement sont mis à 1 (blanc) et ceux qui sont statiques à 0 (noir). Par Conséquent, l'image de fond a été reconstruite avec des valeurs noir et blanc (image binaire voir figure 4.1 (d)).

4.4.3 Implémentation de la méthode proposée

La mise en œuvre de l'approche proposée pour la détection d'objet : estime dans un premier temps les vecteurs de mouvement à l'aide de l'algorithme Diamant en utilisant un seuil adaptatif pour éliminer l'effet de bruit. La deuxième l'étape consiste à binariser l'image. Chaque bloc correspondant à un vecteur de mouvement nul est remplacé par 0 (noir) et les blocs différents sont mis à 1 (blanc) et donc correspondent à un mouvement. Dans la troisième étape de l'algorithme proposé, la morphologie mathématique est utilisée pour réduire le bruit et à assembler les blocs en mouvement. Pour éliminer le bruit du à la binarisation. On a utilisé un filtrage morphologique.

La figure 4.2 présente un exemple qui montre les différentes étapes de l'algorithme proposé pour la détection d'objet en mouvement.



(a) (b)

(c) (d)

Figure 4.2: (a) Champs de mouvement entre deux images successives en utilisant l'algorithme de recherche Diamant [59] avec seuillage (bloc de taille 8x8 pixels et la fenêtre de recherche de taille ± 7 pixels), (b) la trame reconstruite après binarisation, (c) résultat de la détection après l'utilisation de filtrage morphologiques, (d) résultat final de la détection d'objet en mouvement dans la séquence *Sofa*.

Les principales phases de la méthode de détection sont les suivantes :

- Estimation de mouvement entre deux images successives de la vidéo d'entrée en utilisant l'algorithme de recherche Diamant avec seuillage.
- Reconstruction de l'image binaire (les blocs en mouvement sont mis à 1 (blanc) et les blocs statiques sont mis à 0 (noir)).
- Filtrage permettant d'éliminer les pixels isolés ou pour assembler les pixels d'une même région en utilisant des opérations morphologiques
  (Ouverture suivie par fermeture).
- Mise en évidence des objets détectés en les encadrant par des cadres verts autour de centre de gravité d'objets détectés.

4.5 Résultats et discussion.

L'algorithme proposé a été évalué en utilisant la base de données disponible à l'adresse Web :(http ://wordpress-jodoin.dmi.usherb.ca/dataset2012/). Les séquences de test utilisées sont issues d'une caméra fixe, ces séquences contiennent des objets en mouvement(personnes et véhicules ...) et certaines successives images sont sans mouvement. Les résultats de la détection de certaines des séquences de test sont présentés dans les figures ci-dessous. Le temps de traitement dépend de taille de bloc et de changements qui se produisent dans un arrière-plan. Pour un fond stable avec quelques blocs détectés, la complexité de l'algorithme est relativement petite. À travers les résultats de test on constate que l'algorithme arrive à détecter les objets en mouvement sur toutes les vidéos de test. Nous pouvons voir que notre méthode proposée peut effectivement détecter des objets en mouvement avec différentes tailles et formes en temps réel.

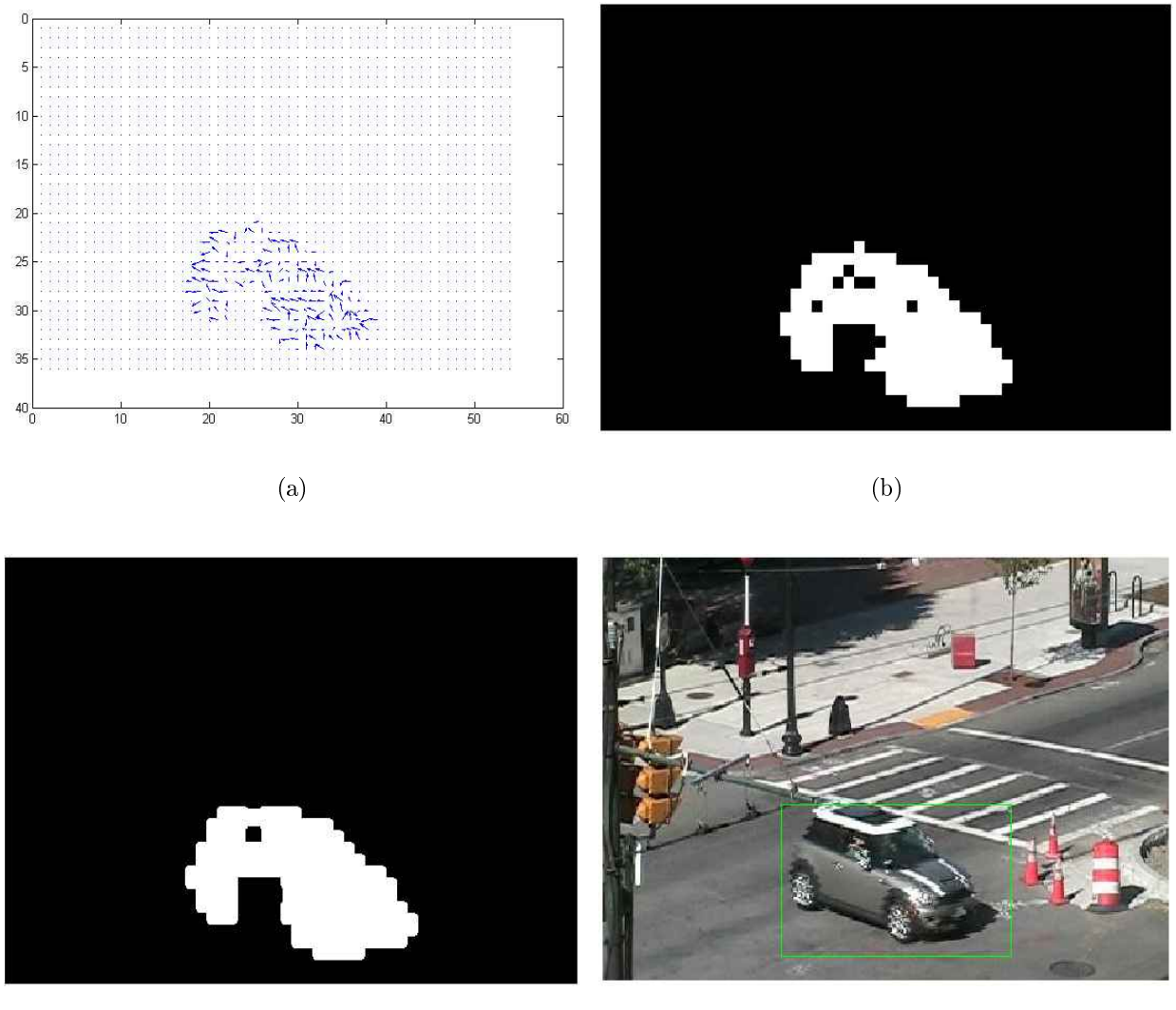


Figure 4.3: (a) Champs de mouvement entre deux images successives en utilisant l'algorithme de recherche Diamant [59] avec seuillage (bloc de taille 8x8 pixels et la fenêtre de recherche de taille ± 7 pixels), (b) la trame reconstruite après binarisation, (c) résultat de la détection après l'utilisation de filtrage morphologique, (d) résultat final de la détection d'objet en mouvement dans la séquence *Abandoned Box*.

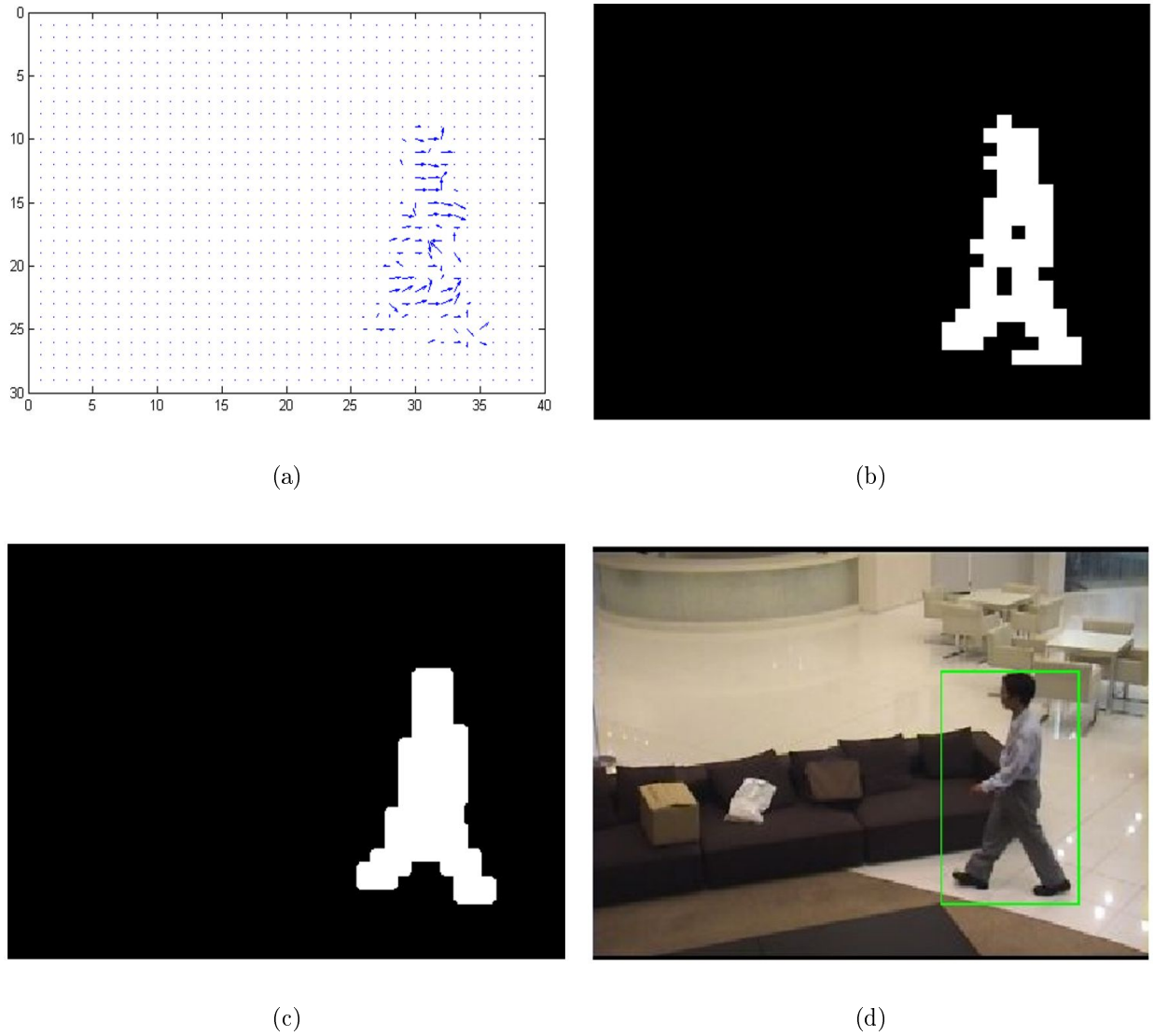


Figure 4.4: (a) Champs de mouvement entre deux images successives en utilisant l'algorithme de recherche Diamant [59] avec seuillage (bloc de taille 8x8 pixels et la fenêtre de recherche de taille ± 7 pixels), (b) la trame reconstruite après binarisation, (c) résultat de la détection après l'utilisation de filtrage morphologique, (d) résultat final de la détection d'objet en mouvement dans la séquence *Sofa*.

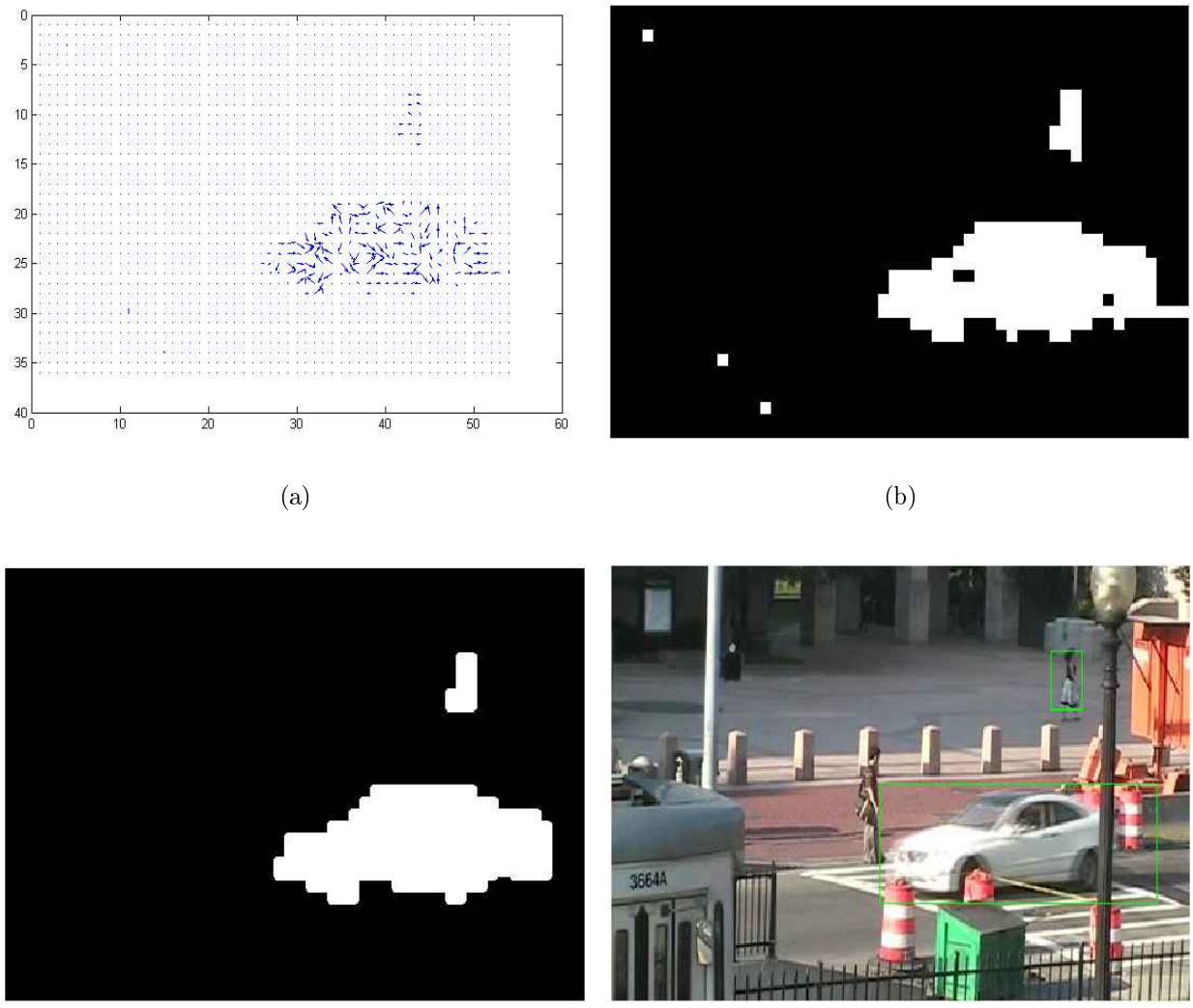


Figure 4.5: (a) Champs de mouvement entre deux images successives en utilisant l'algorithme de recherche Diamant [59] avec seuillage (bloc de taille 8x8 pixels et la fenêtre de recherche de taille ± 7 pixels), (b) la trame reconstruite après binarisation, (c) résultat de la détection après l'utilisation de filtrage morphologique, (d) résultat final de la détection d'objet en mouvement dans la séquence *Tram Stop*.

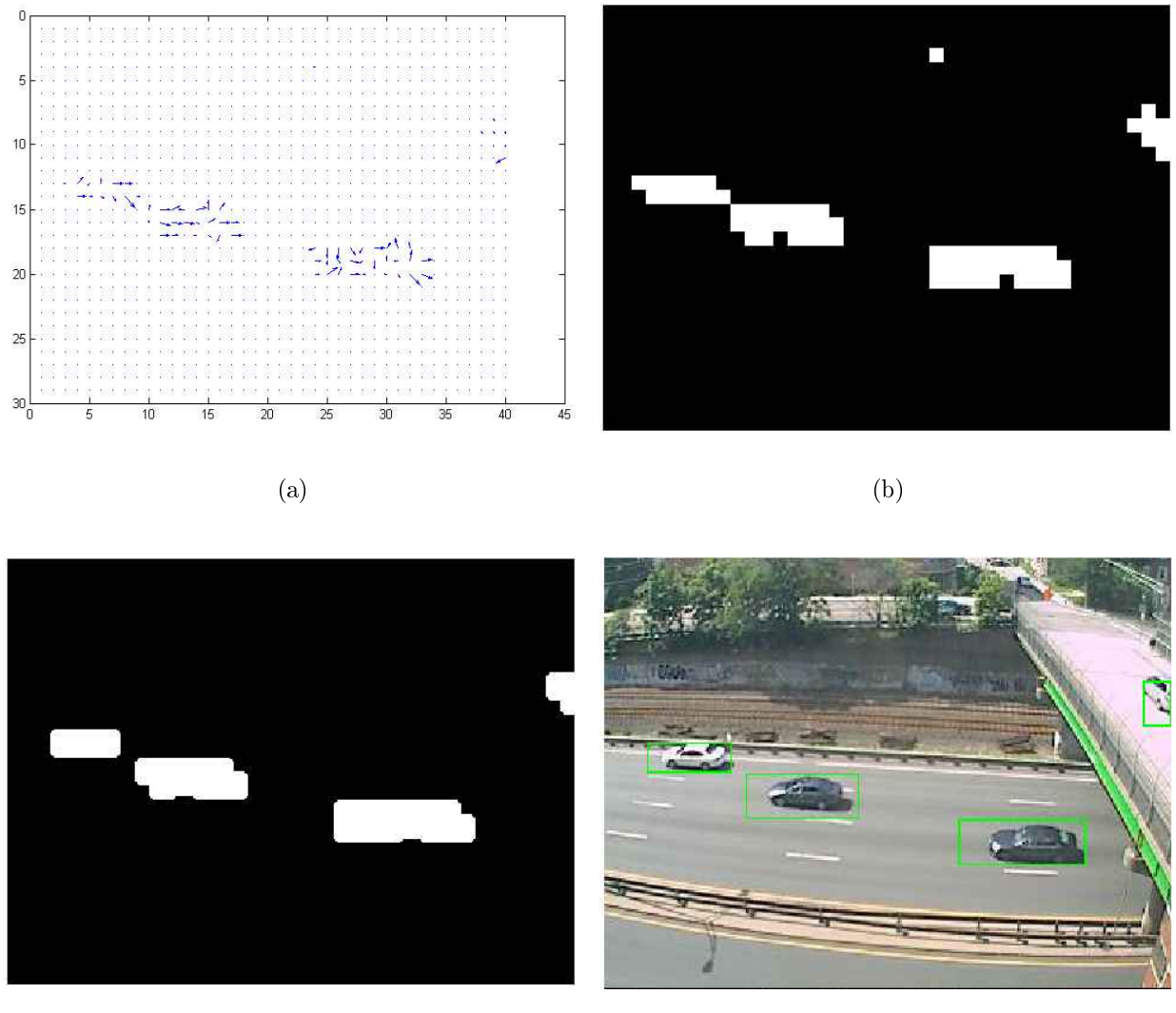


Figure 4.6: (a) Champs de mouvement entre deux images successives en utilisant l'algorithme de recherche Diamant [59] avec seuillage (bloc de taille 8x8 pixels et la fenêtre de recherche de taille ± 7 pixels), (b) la trame reconstruite après binarisation, (c) résultat de la détection après l'utilisation de filtrage morphologique, (d) résultat final de la détection d'objet en mouvement dans la séquence *Street Light*.

Quelques résultats sont présentés dans le tableau suivant en comparant le temps de calcul (en second) entre notre algorithme proposé et l'algorithme de MoG [79] qui donne les mêmes résultats visuel. Les résultats sont obtenus à l'aide d'un PC Intel Core i7-3770 CPU @ 3.40GHz 3.40 GHz avec 8 Go de RAM et (3.41 Go utilisable).

Liste des tableaux 4.1: Comparaison de temps de calcul (en secondes) de l'algorithme proposé et l'algorithme de MoG.

Séquences Méthodes	**Sofa**	***Abandoned Box***	***Street light***	***Tram stop***
MoG	3.12	4.31	3.85	4.54
Algo Proposé	0.57	0.63	0.60	0.67
Amélioration(%)	81.73	85.38	81.53	

On remarque bien que l'algorithme proposé réduit de manière remarquable environ 84% de temps de calcul sur toutes les séquences comparativement à l'algorithme MoG, ce qui permet temps réel. Les expériences réalisées ont montré que l'algorithme arrive à détecter la plupart d'objets en mouvement sur toutes les scènes simple et complexe. Et la détection d'objets en mouvement est réalisée en temps réel. Cependant dans le cas où l'objet se déplace très lentement, il est intégré dans le fond. Par conséquent, dans ce cas-là, l'objet est mal détecté.

4.6 Conclusion

Nous avons présenté dans cette partie un algorithme conçu pour détecter les objets mobiles. L'idée de base du l'algorithme proposé consiste, en utilisant le champ de mouvement, à mettre les blocs de l'image en mouvement à 1 les blocs statiques à 0. L'algorithme de recherche Diamant avec seuil a été utilisé. La mise en forme de l'image binarisée a été réalisée grâce à un filtre morphologique. L'algorithme proposé comprend les étapes suivantes : la première consiste à estimer les vecteurs de mouvement entre les trames successives en utilisant l'algorithme de recherche en Diamant, dans la seconde étape, les blocs en mouvement sont remplacés avec des blocs blanc et les blocs statiques sont remplacés par des valeurs nulles (bloc noir). Dans

la troisième étape, les filtres de morphologique mathématique (Ouverture et Fermeture) sont utilisés pour l'affinage des objets détectés. Par conséquent, nous pouvons dire que les objets en mouvement ont été détectés à partir de l'arrière-plan statique. Les avantages de l'algorithme proposé sont la simplicité de mise en œuvre, la rapidité et la précision de détection en temps réel. Cependant, le choix de la taille de bloc aura un grand impact sur les résultats et dans le cas où l'objet se déplacerait lentement, il est inclut dans le fond. Un ensemble d'exemple sont donnés montrant ainsi l'efficacité de l'algorithme que nous proposons.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Cette thèse a été dédiée aux méthodes d'estimation de mouvement permettent d'améliorer les techniques de compression et de codage vidéo. Ceci pour assurer une bonne qualité vidéo et une complexité réduite. La première partie de notre travail a consisté à faire un état de l'art des méthodes existants dans la littérature scientifique et à étudier les différents algorithmes d'estimation de mouvement. Nous avons effectué une étude approfondie des algorithmes d'estimation du mouvement ainsi que les modèles de recherche employés.

La méthode Full Search (FS) effectue une recherche exhaustive et assure l'optimalité de la solution cependant elle est d'une trop grande complexité. Afin de la réduire plusieurs méthodes ont été élaborées. La plus connue est la méthode Diamond Search (DS). Celle-ci effectue la recherche à l'aide deux modèles. Le premier, appelé LDSP (Large Diamond Search Pattern) calcule neuf points alors que le second SDSP (Small Diamond Search Pattern) calcule cinq points. L'algorithme DS, comparativement aux autres algorithmes est performant du point de vue complexité/qualité. Il est bien adapté aux petits mouvements (≤ 2 pixels), cependant il est de moindre efficacité pour les mouvements de grande amplitude.

(S. Zhu and K. K. Ma, 2000) montrent que la distribution de la fréquence du vecteur de mouvement en fonction de son amplitude est à 80% des cas à l'intérieur d'un disque de rayon égal à 2 pixels . Ceux-ci affirment aussi que sa fréquence est relativement élevée sur les axes horizontal

xx et vertical yy sans donner aucune information quantitative. Afin d'élargir notre connaissance sur sa distribution nous avons effectué une étude de la distribution de la fréquence du vecteur mouvement sur un corpus de 12 vidéos composées chacune de 150 images.
Nous proposons une nouvelle approche (dénommée ED) qui permet d'améliorer les performances de Diamond Search en dehors du disque de rayon 2 pixels. On a vu que l'algorithme proposé ED exploite le fait que la fréquence du vecteur de mouvement est élevée sur les axes xx et yy (ce qui correspond à des déplacements horizontaux ou verticaux respectivement). ED procède en deux étapes : la première recherche suit une forme d'étoile (effectue une recherche grossière), une forme de Petit Diamant affine le résultat de la recherche. La notion de nombre de points calculés NPC est défini pour la première fois dans le présent document. NPC est défini pour le bloc et NPCmoy pour la séquence d'image. Nous pensons que ce paramètre est plus utilisé pour mesurer la complexité d'un algorithme. Une étude comparative avec les méthodes standards FS, TSS, 4SS, NOCDS et en particulier la méthode Diamant a montré que l'algorithme proposé Étoile Diamant est moins complexe que Diamant : il y a une amélioration de 7,86% en moyenne sur l'ensemble du corpus utilisé dans le présent document (12 séquences de 150 images). Il a également été montré que pour des séquences rapides (bus) l'amélioration de pourcentage pourrait atteindre 14,85%. Pour une estimation plus rapide nous avons remarqué à partir de l'étude statistique que et 50.06% des vecteurs de mouvement sont nuls et qu'on peut utiliser un seuil adaptatif défini à partir de l'erreur de mise en correspondance de bloc pour éliminer les blocs statiques durant l'estimation de mouvement, ce qui diminue remarquablement le nombre de points calculés et donc la complexité. L'étude comparative, montre que notre méthode réduit d'environ (30-70%), le nombre de points calculés par bloc avec une même qualité de l'image par rapport aux autres algorithmes. Les algorithmes proposés s'apprêtent mieux aux applications de codage vidéo en temps réel.

La deuxième partie est consacrée à la détection des objets en mouvement à partir d'une séquence vidéo qui est l'élément clé de nombreuses applications de vision tels que la vidéo surveillance, le suivi du trafic, la réalité augmentée, la navigation automobile, la télévision ainsi que des services de télécommunications. L'efficacité et la précision de ces applications dé-

pendent des résultats issus des méthodes de la détection utilisées. De nombreuses méthodes de la détection d'objet en mouvement existent dans la littérature scientifique . Nous classons ces méthodes en trois types : Méthodes basées sur la distribution, méthodes de flot optique et les méthodes basées contour. Notre contribution porte sur la proposition d'une nouvelle méthode de détection des objets en mouvement on utilisant l'estimation et la compensation de mouvement à l'aide de l'algorithme Diamant avec seuillage et de la morphologie mathématique. La méthode proposée comprend les étapes suivantes : à estimer les vecteurs de mouvement entre des trames successives utilisant l'algorithme Diamant, puis la binarisation blocs statiques en noir et blocs dynamiques en blanc. Les filtres de la morphologie mathématique (ouverture et fermeture) sont utilisés pour une meilleure détection de l'objet en mouvement. L'efficacité de l'algorithme proposé est dans la simplicité de mise en œuvre et la précision de la détection en temps réel. Cependant, le choix de la taille de bloc a un grand impact sur les résultats. L'efficacité de notre approche est illustrée par un ensemble d'exemples.

Perspectives

Au moment de clore ce manuscrit, des perspectives prometteuses apparaissent suite aux travaux réalisés. Les principales voies de développement possibles sont l'implémentation des algorithmes proposées dans un environnement conjoint matériel/logiciel du nouveau standard de compression vidéo HEVC ainsi que l'incorporation dans la plate-forme de compression vidéo pour des applications de vidéo surveillance, de mesures temps réel tels que le suivi d'objets ou de la détection d'événements, la reconnaissance de formes. Dans ces applications, la qualité de la vidéo obtenue influence sur la fiabilité de ces derniers par exemple apparition de l'objet dans l'image. Un individu (objet) est localisé dans la scène, il est possible de le suivre (tracking) mais aussi de compresser les données de manière spécifique en fonction de la région ou le temps de la partition afin, par exemple, de pouvoir le reconnaître (les données des objets (zone d'intérêt) étant moins compressées que le reste de la scène).

BIBLIOGRAPHIE

[1] Mohammed Ghanbari. *Standard codecs : Image compression to advanced video coding.* Number 49. Iet, 2003.

[2] Abdelazim Abdelrahman. *Fast Motion Estimation Algorithms for BlockBased Video Coding Encoders.* PhD thesis, Applied Digital Signal and Image Processing Research Centre, School of Computing, Engineering and Physical Sciences, University of Central Lancashire, 2011.

[3] Jörn Ostermann, Jan Bormans, Peter List, Detlev Marpe, Matthias Narroschke, Fernando Pereira, Thomas Stockhammer, and Thomas Wedi. Video coding with h. 264/avc : tools, performance, and complexity. *Circuits and Systems magazine, IEEE*, 4(1) :7–28, 2004.

[4] Benmoussat Nawal. *Contributions To Motion Estimation,Compensation and Temporal Interpolation Techniques For Video Transmission.* PhD thesis, USTO-MB, 2005.

[5] Anouar Hadj TAEIB, Imen WERDA, Amine SAMET, Mohamed Ali BEN AYED, and Nouri MASMOUDI. Apport en qualité du processus de filtrage appliqué a la norme de codage vidéo h. 264.

[6] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *Image Processing, IEEE Transactions on*, 9(2) :287–290, 2000.

[7] Iain E Richardson. *Video codec design : developing image and video compression systems.* John Wiley & Sons, 2002.

[8] Kerfa Djoudi and Belbachir M.F. Star diamond : an efficient algorithm for fast block matching motion estimation in h264/avc video codec. *Multimedia Tools and Applications*, pages 1–15, Springer,2015.

[9] Kerfa Djoudi et Belbachir M.F. Etoile-diamant : un nouvel algorithme efficace pour l'estimation de mouvement par la mise en correspondance de blocs. *8ème Ateliers de Traitement et Analyse de l'Information : Méthodes et Applications «TAIMA'13 »*, Hammamet(Tunisie.(session orale)) :ISBN : 978–9938–05–715–7., du 13 au 18 Mai 2013,.

[10] Kerfa Djoudi and M.F Belbachir. An efficient real time moving object detection scheme using diamond search algorithm and mathematical morphology. *International Review on Computers and Software*, 9(5) :744–749, 2014.

[11] Kerfa Djoudi and Belbachir Mohamed.F. A new moving object detection method for visual surveillance. *IT4OD*, page 223, 2014.

[12] Kerfa Djoudi and Belbachir Mohamed F. An efficient algorithm for fast block matching motion estimation. *MISC*, 2014.

[13] Ioannis Pitas. *Digital image processing algorithms and applications.* John Wiley & Sons, 2000.

[14] Gabriel Peyré. Le traitement numérique des images. CNRS. 2011. <hal-00690096>.

[15] Antoine Robert. *Transformées orientées par blocs pour le codage vidéo hybride.* PhD thesis, Télécom ParisTech, 2008.

[16] Alan C Bovik. *Handbook of image and video processing.* Academic Press, 2010.

[17] Crespin Sébastien. mpeg website. `http://screspin.free.fr/mpeg/`, 5 décembre 2001.

[18] Hong Man, Alen Docef, and Faouzi Kossentini. Performance analysis of the jpeg 2000 image coding standard. *Multimedia Tools and Applications*, 26(1) :27–57, 2005.

[19] Guy Côté and Lowell Winger. Progrès récents dans le domaine de la compression vidéo. *IEEE canadian Review*, 2002.

[20] E. Incerti. *Compression d'image : algorithmes et standards.* Vuibert, 2003.

[21] Maria Trocan. *Décompositions spatio-temporelles et allocation de débit utilisant les coupures des graphes pour le codage vidéo scalable.* PhD thesis, Télécom ParisTech, 2007.

[22] Karlheinz Brandenburg and Gerhard Stoll. Iso/mpeg-1 audio : A generic standard for coding of high-quality digital audio. *Journal of the Audio Engineering Society*, 42(10) :780–792, 1994.

[23] Thomas Sikora. The mpeg-4 video standard verification model. *Circuits and Systems for Video Technology, IEEE Transactions on*, 7(1) :19–31, 1997.

[24] Bangalore S Manjunath, Philippe Salembier, and Thomas Sikora. *Introduction to MPEG-7 : multimedia content description interface*, volume 1. John Wiley & Sons, 2002.

[25] I. T. U. T. H. 261 REC. Rec, i. t. u. t. h. 261,video codec for audio visual service atp. 64, 1993.

[26] Karel Rijkse. H. 263 : video coding for low-bit-rate communication. *Communications Magazine, IEEE*, 34(12) :42–45, 1996.

[27] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7) :560–576, 2003.

[28] Guy Cote, Berna Erol, Michael Gallant, and Faouzi Kossentini. H. 263+ : Video coding at low bit rates. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(7) :849–866, 1998.

[29] Stéphane PÉCHARD, Patrick Le Callet, and Vincent Ricordel. Utilisation de modèles psychovisuels pour le contrôle de la prédiction spatiale de la norme de compression vidéo h. 264/avc. 2004.

[30] Fabrice Urban, Jean François Nezan, Olivier Déforges, Ronan Poullaouec, et al. Estimateur de mouvement temps réel multi-dsp pour l'encodage vidéo mpeg-4 avc/h. 264 haute définition. *COmpression et REprésentation des Signaux Audiovisuels (CORESA)*, 2006.

[31] Majdi Elhajji. *Co-Design de l'application H264 et implantation sur un NoC-GALS*. PhD thesis, Lille 1, 2012.

[32] Chodisetti Rakesh Anil. H. 264 video coding artifacts. 2013.

[33] Miryem Hrarti. *Optimisation du contrôle de débit de H. 264/AVC basée sur une nouvelle modélisation Débit-Quantification et une allocation sélective de bits.* PhD thesis, Poitiers, 2011.

[34] Olivier Brouard. *Pré-analyse de la vidéo pour un codage adapté. Application au codage de la TVHD en flux H. 264.* PhD thesis, Université de Nantes, 2010.

[35] Cédric Marin. *Vers une solution réaliste de décodage source-canal conjoint de contenus multimédia.* PhD thesis, Université Paris Sud-Paris XI, 2009.

[36] Moez Kthiri. *Étude et implantation d'algorithmes de compression vidéo optimisés H. 264/AVC dans un environnement conjoint matériel et logiciel.* PhD thesis, Bordeaux 1, 2012.

[37] Camille Mazataud. Error concealment for h. 264 video transmission. 2009.

[38] Nikolay Ponomarenko, Flavia Silvestri, Karen Egiazarian, Marco Carli, Jaakko Astola, and Vladimir Lukin. On between-coefficient contrast masking of dct basis functions. In *Proceedings of the Third International Workshop on Video Processing and Quality Metrics*, volume 4, 2007.

[39] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment : from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4) :600–612, 2004.

[40] Yunju Baek, Hwang-Seok Oh, and Heung-Kyu Lee. Block-matching criterion for efficient vlsi implementation of motion estimation. *Electronics Letters*, 32(13) :1184–1185, 1996.

[41] T Koga. Motion-compensated interframe coding for video conferencing. In *Proc. NTC 81*, pages C9–6, 1981.

[42] RA Manap, SSS Ranjit, Amat Amir Basari, and Badrul Hisham Ahmad. Performance analysis of hexagon-diamond search algorithm for motion estimation. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 3, pages V3–155. IEEE, 2010.

[43] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

[44] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3) :433–466, 1995.

[45] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1) :43–77, 1994.

[46] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[47] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5, 2001.

[48] Joseph Weber and Jitendra Malik. Robust computation of optical flow in a multi-scale differential framework. *International Journal of Computer Vision*, 14(1) :67–81, 1995.

[49] David J Fleet and Allan D Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1) :77–104, 1990.

[50] Yu-Te Wu, Takeo Kanade, Jeffrey Cohn, and C-C Li. Optical flow estimation using wavelet motion model. In *Computer Vision, 1998. Sixth International Conference on*, pages 992–998. IEEE, 1998.

[51] Vasileios Argyriou and Theodore Vlachos. A study of sub-pixel motion estimation using phase correlation. In *BMVC*, pages 387–396. Citeseer, 2006.

[52] H Gharavi and Mike Mills. Blockmatching motion estimation algorithms-new results. *Circuits and Systems, IEEE Transactions on*, 37(5) :649–651, 1990.

[53] Xiaoyin Cheng. Subjectively optimized hdtv video coding. 2009.

[54] Siwei Ma, Wen Gao, and Yan Lu. Rate-distortion analysis for h. 264/avc video coding and its application to rate control. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(12) :1533–1544, 2005.

[55] Yu-Wen Huang, Bing-Yu Hsieh, Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen. Analysis and complexity reduction of multiple reference frames motion estimation in h. 264/avc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 16(4) :507–522, 2006.

[56] Iain E Richardson. *H. 264 and MPEG-4 video compression : video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[57] Renxiang Li, Bing Zeng, and Ming L Liou. A new three-step search algorithm for block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 4(4) :438–442, 1994.

[58] Jianhua Lu and Ming L Liou. A simple and efficient search algorithm for block-matching motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 7(2) :429–433, 1997.

[59] Lai-Man Po and Wing-Chung Ma. A novel four-step search algorithm for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(3) :313–317, 1996.

[60] Chun-Ho Cheung and Lai-Man Po. A novel cross-diamond search algorithm for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(12) :1168–1177, 2002.

[61] Ce Zhu, Xiao Lin, and L-P Chau. Hexagon-based search pattern for fast block motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(5) :349–355, 2002.

[62] Xuzhi Wang, Wanggen Wan, Jinyuan Zhang, and Yanru Ma. Research on the motion estimation with a novel octagon cross diamond search algorithm. In *Microelectronics and Electronics (PrimeAsia), 2010 Asia Pacific Conference on Postgraduate Research in*, pages 89–92. IEEE, 2010.

[63] Luc De Vos and Michael Stegherr. Parameterizable vlsi architectures for the full-search block-matching algorithm. *Circuits and Systems, IEEE Transactions on*, 36(10) :1309–1316, 1989.

[64] Video test media. `http://media.xiph.org/video/derf/`.

[65] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Computer Vision–ECCV 2006*, pages 14–28. Springer, 2006.

[66] Rongbo Zhu. Intelligent collaborative event query algorithm in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2012, 2011.

[67] Tao Yang, Stan Z Li, Quan Pan, and Jing Li. Real-time and accurate segmentation of moving objects in dynamic scene. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, pages 136–143. ACM, 2004.

[68] Yair Weiss and Edward H Adelson. A unified mixture framework for motion segmentation : Incorporating spatial coherence and estimating the number of models. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 321–326. IEEE, 1996.

[69] Jean-Marc Odobez and Patrick Bouthemy. Direct incremental model-based image motion segmentation for video analysis. *Signal Processing*, 66(2) :143–155, 1998.

[70] Thomas Brox, Andrés Bruhn, and Joachim Weickert. Variational motion segmentation with level sets. In *Computer Vision–ECCV 2006*, pages 471–483. Springer, 2006.

[71] Masayuki Yokoyama and Tomaso Poggio. A contour-based moving object detection and tracking. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 271–276. IEEE, 2005.

[72] Wen Fang and Kap Luk Chan. Using statistical shape priors in geodesic active contours for robust object detection. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 304–307. IEEE, 2006.

[73] Alan A Stocker. An improved 2d optical flow sensor for motion segmentation. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 2, pages II–332. IEEE, 2002.

[74] Surya PN Singh, Paul J Csonka, and Kenneth J Waldron. Optical flow aided motion estimation for legged locomotion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1738–1743. IEEE, 2006.

[75] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[76] Liyuan Li, Weimin Huang, Irene YH Gu, and Qi Tian. Foreground object detection in changing background based on color co-occurrence statistics. In *Applications of Computer*

Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on, pages 269–274. IEEE, 2002.

[77] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-Based Surveillance Systems*, pages 135–144. Springer, 2002.

[78] Rongbo Zhu. Efficient fault-tolerant event query algorithm in distributed wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2010, 2010.

[79] Rudolf Mester, Til Aach, and Lutz Dümbgen. Illumination-invariant change detection using a statistical colinearity criterion. In *Pattern Recognition*, pages 170–177. Springer, 2001.

[80] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *Computer Vision—ECCV 2000*, pages 751–767. Springer, 2000.

[81] H Askar and Xiaofeng Li. Background clutter suppression and dim moving point targets detection using nonparametric method. In *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on*, volume 2, pages 982–986. IEEE, 2002.

[82] David Thirde and Graeme Jones. Hierarchical probabilistic models for video object segmentation and tracking. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 636–639. IEEE, 2004.