N° d'ordre :

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DJILLALI LIABES
FACULTE DES SCIENCES EXACTES
SIDI BEL ABBES

THESE

DE DOCTORAT

Présentée par

ELBAHRI MOHAMED

Spécialité : Informatique
Option : Traitement d'images

Intitulé

A highly efficient algorithm for fast motion detection and estimation using parallel processing

Soutenue le……

Devant le jury composé de :

Président : M. YOUSFATE Abderrahmane Pr. UDL, Sidi bel Abbes
Examinateurs : M. RAHMOUN Abdellatif Pr. ESI, Sidi bel Abbes
M. CHIKH Mohamed El Amine Pr. univ e Tlemcen
M. KPALMA Kidiyo Pr. INSA de Rennes, France
M. FARAOUN Kamel Mohamed, Pr. UDL, Sidi bel Abbes
Invité: M. RONSIN Joseph Pr. INSA de Rennes, France

Directeur de thèse : M. TALEB Nasreddine Pr. UDL, Sidi bel Abbes

Remerciements

Ce travail a été réalisé sous la direction de Monsieur le Professeur ***Nasredinne TALEB.***

Je souhaite exprimer toute ma gratitude et mes profonds remerciements à mon encadreur Pr. ***N. TALEB*** *pour ses généreux conseils, ses encouragements et son aide très précieuse. Sa disponibilité, sa grande culture scientifique, sa rigueur avec laquelle il aborde un problème scientifique, son ouverture d'esprit et sa compétence m'ont beaucoup permis d'élargir ma vision sur ce sujet de recherche.*

Je suis particulièrement reconnaissant au Pr. ***Kidiyo KPALMA*** *qui a accepté de m'accueillir au sein du laboratoire IETR de l'INSA de Rennes (FRANCE) et qui a contribué à la réalisation de ce travail. Qu'il me soit permis de le remercier vivement pour sa grande disponibilité et sa bienveillante attention.*

Mon profond respect et mes vifs remerciements sont adressés au Pr. ***YOUCEFATE Abderrahmane****, d'avoir accepté de présider ce jury.*

Je tiens à remercier Messieurs ***Pr. RAHMOUN Abdellatif, Pr. CHIKH Mohamed El Amine et Pr. FARAOUN Kamel Mohamed*** *pour avoir mobilisé de leur temps afin d'examiner et de juger ce travail. Qu'ils trouvent ici toute ma reconnaissance et ma gratitude.*

Mes vifs remerciements sont aussi adressés au ***Pr. RONSIN Joseph*** *du laboratoire IETR de l'INSA de Rennes (FRANCE) pour son aide précieuse et ses prodigieux conseils.*

Je suis très heureux de témoigner ma sympathie aux responsables et enseignants de notre faculté.

بســـم الله الـــرحمن الـــرحيم

AU NOM D'ALLAH LE MISERICORDIEUX

Je dédie ce travail à

Mes très chers parents

et

Mes sœurs

Résumé

La surveillance visuelle des scènes dynamiques, est l'un des difficiles sujets de recherche actuels en vision par ordinateur. C'est une technologie clé pour lutter contre le terrorisme, la criminalité, pour assurer la sécurité publique et pour la gestion efficace de la circulation routière. Ce travail consiste à concevoir un système efficace de détection et d'estimation de mouvement pour le suivi d'objet dans un environnement complexe. Pour la surveillance vidéo, la détection d'objets en mouvement à partir d'une vidéo est importante pour la classification d'objet, suivi de cible, la reconnaissance de l'activité, et la compréhension du comportement. Dans cette thèse nous avons utilisé la soustraction du fond pour la détection du mouvement. Pour pouvoir reconnaitre les objets, nous avons utilisé un système de classification basé sur les représentations parcimonieuses. Une accélération du traitement est proposée dans ce travail, en utilisant les performances des cartes graphiques. Cette accélération consiste à paralléliser l'algorithme "Othogonal Matching Pursuit" pour accélérer le traitement et le rendre opérant en temps-réel.

Abstract

Visual surveillance of dynamic scenes is one of the difficult current research topics in computer vision. It is a key technology in the fight against terrorism, crime, to ensure public safety and to the effective management of road traffic. This work is to design an effective system of detection and motion estimation for object tracking in a complex environment. In video surveillance, detection of moving objects from a video is important for objects classification, target tracking, recognition of the activity, and understanding behavior. In this thesis we used background subtraction for motion detection. In order to recognize objects, we used a classification system based on sparse representation. A faster processing is proposed in this work, using the performance of graphics cards. This acceleration consists to parallelize the "Othogonal Matching Pursuit" algorithm to speed up the processing and make it operate in real-time.

ملخص:

المراقبة بالفيديو للمشاهد الحيوية، هي واحدة من أصعب الموضوعات البحثية الحالية في الرؤية الحاسوبية، هذه التكنولوجيا مهمة في مجال مكافحة الإرهاب والجريمة، لضمان السلامة العامة والإدارة الفعالة لحركة المرور على الطرق. هذا العمل هو تصميم نظام فعال للكشف وتقدير الحركة لتتبع الكائن في بيئة معقدة. في المراقبة بالفيديو، والكشف عن الأجسام المتحركة من الفيديو مع تصنيف الكائن، وتتبع الهدف، وتقدير النشاط، في هذه الأطروحة استخدمنا الطرح الخلفية لكشف الحركة. من أجل التعرف على الأشياء، استخدمنا نظام التصنيف على أساس التمثيل المتفرق. ويقترح معالجة أسرع في هذا العمل، وذلك باستخدام أداء بطاقة العرض المرئي. هذا التسارع هو توازي للخوارزمية المستعملة. لتسريع النظام وجعله يعمل في الوقت الحقيقي.

Table des matières

Introduction générale

De nos jours, l'imagerie numérique touche tous les secteurs: sécurité, industrie, médecine et encore d'autres domaines. Avec la généralisation de l'utilisation des images numériques, un nouveau concept a surgi, c'est la vision par ordinateur, appelée aussi vision artificielle, qui n'est autre que la combinaison entre deux axes techniques: le traitement d'image et l'intelligence artificielle. La vision par ordinateur est définie par les théoriciens Radu Horaud et Olivier Monga de la manière suivante : « La vision est un processus de traitement de l'information. Elle utilise des stratégies bien définies afin d'atteindre ses buts. L'entrée d'un système de vision est constituée par une séquence d'images. Le système lui-même apporte un certain nombre de connaissances qui interviennent à tous les niveaux. » Parmi les nombreuses applications en vision par ordinateur, on cite la surveillance vidéo, l'interaction homme machine, la robotique, l'imagerie médicale, la compression vidéo .... En effet toutes ces applications nécessitent un traitement qui est l'analyse du mouvement dans une séquence vidéo. L'analyse du mouvement dans l'espace tridimensionnel est une tâche banale dans le monde animal et humain alors qu'il s'est avéré très difficile d'obtenir une capacité visuelle sur une machine, identique à une autre qui est naturelle, même avec un degré très limité et une variété très large de détecteurs de mouvement et de systèmes visuels, et des calculateurs puissants. Dans cette thèse, on utilisera l'analyse et l'estimation du mouvement dans des scènes encombrées pour suivre et tracer le parcours de chaque objet en mouvement.

Première problématique

Dans une séquence d'images, plusieurs objets bougent. Comment peut-on créer un système capable de les reconnaitre et tracer le mouvement de chacun d'eux? Pour analyser un mouvement dans ce genre de situation, on doit se consacrer à résoudre un certain nombre de problématiques:

- La détection du mouvement: chaque pixel de l'image doit être désigné comme appartenant au fond qui est fixe ou appartenant à une zone de mouvement.
- La détection des objets en mouvement, autrement dit: séparer l'avant plan de l'arrière plan.
- La segmentation basée région: distinguer chaque région selon son mouvement.

- La classification des objets: Les objets détectés dans une scène peuvent être des personnes, des animaux, des véhicules, des arbres secoués par le vent... Il est essentiel de pouvoir distinguer entres ces derniers. Même dans des scènes ne contenant que des personnes, il faut bien reconnaitre chacune.
- Le suivi d'objet: déterminer à chaque instant la position de cet objet dans la scène.
- La mesure du mouvement: appelée aussi estimation de mouvement, qui consiste à estimer le mouvement apparent et à déterminer le vecteur mouvement.

Pour concevoir un système de suivi de personne fiable, il faudra aborder les six points cités ci-dessus, c.-à-d. qu'il est impératif de passer par l'étape de détection du mouvement, détection des objets et segmentation basée mouvement, classification, suivi et enfin estimer le mouvement de chaque objet.

Seconde problématique

Un système d'analyse de mouvement risque d'être lent et lourd à cause du volume d'information codant la vidéo, et l'ensemble des opérations nécessaires au traitement de ces données, alors qu'on exige qu'il soit un système temps-réel. Des solutions sont proposées dans des travaux ultérieurs de l'état de l'art, mais la plus part d'eux, diminuent la précision de la détection pour accélérer le système; Alors que le challenge est de maintenir la précision de la détection de chaque objet en mouvement en temps-réel.

Contributions

Avant de définir les contributions de notre travail, définissons le besoin et les finalités de ce travail. Imaginons qu'on est dans un balcon ou un endroit qui domine une vue, dans cette scène observée, il y a beaucoup de personnes qui traversent la rue, ou marchent dans un endroit public. Est-il possible de tracer et suivre le mouvement de chaque personne faisant partie de cette scène observée ? je pense que tout le monde est d'accord que la réponse est bien "c'est une tâche difficile, voir impossible". Ce travail est consacré à la création d'un système permettant l'analyse du mouvement dans ce genre de situation. Pour atteindre notre but, nous nous sommes basés sur les représentations parcimonieuses pour la classification des objets via un système d'apprentissage. Et comme autre contribution, on propose une accélération du traitement en utilisant les cartes graphiques. Ces cartes graphiques appelées (GPU) ont été créées pour accélérer des traitements graphiques (les jeux vidéo). Maintenant,

on les considère comme des systèmes massivement parallèles avec lesquels, le paradigme "Single instruction multi threading: SIMT" peut être implémenté très facilement.

Le système conçu dans ce travail est composé de quatre étapes essentielles:

- La détection des objets en mouvement
- La représentation de ces objets
- La reconnaissance de ces objets
- L'estimation du mouvement de chaque objet

Nous proposons la soustraction du fond comme méthode de détection pour sa rapidité et sa facilité d'implémentation, mais ça n'empêche pas que d'autres méthodes plus complexes peuvent être prposées pour extraire l'avant plan (les objets en mouvement) sur ce genre de scènes.

Après une segmentation basée sur le mouvement, chaque objet détecté est représenté par un descripteur contenant des caractéristiques basées sur son apparence et sa position dans la scène.

Pour reconnaitre et distinguer chaque objet des autres, un dictionnaire contenant des sous classes est utilisé. Chacune de ces sous-classes représente un objet déjà reconnu et étiqueté. Si un objet apparait pour la première fois, une sous-classe est ajoutée au dictionnaire, sinon sa classe sera mise à jour. Un algorithme appelé "Orthogonal Matching Pursuit" fut utilisé pour générer le vecteur de parcimonie, dont les coefficients sont utilisés pour la classification de chaque objet.

Une fois chaque cible détectée est classifiée, on peut facilement tracer son parcours et estimer son mouvement.

Notons que nous nous sommes concentrés sur des scènes encombrées de personnes pour valider l'efficacité de notre approche, et pour pouvoir comparer nos résultats avec d'autres de l'état de l'art utilisant le même type de bases de données référence.

Chapitre I

Etat de l'art sur la détection et l'estimation du mouvement

I. 1 Introduction

La détection des régions correspondants aux objets en mouvement tels que des véhicules, des personnes ou autre objets dans des scènes naturelles est un problème important et difficile, qui fournit un grand intérêt pour la réalisation des autres étapes ultérieures d'analyse et de traitement de la vidéo. La segmentation et la détection du mouvement est un problème ouvert et difficile en raison des changements dynamiques dans les scènes naturelles tels que les changements d'éclairage globales et locales, l'occlusion, le déplacement répétitif des objets par exemple: l'agitation brusque des drapeaux, des rubans, ou les feuilles des arbres; et pour d'autres raison physique de changements de la scène.

Dans ce chapitre nous définissons le mouvement ainsi que ses différents types, selon son emplacement et la manière qui permet de l'estimer. Il existe trois type essentiel de mouvement: le mouvement réel, le mouvement apparent et le mouvement estimé. Par la suite nous présenterons les différentes méthodes de détection et d'estimation de mouvement qui existe dans l'état de l'art. Dans notre travail nous avons utilisé une méthode simple pour la détection de mouvement sans une amélioration, par contre, pour pouvoir améliorer la précision de l'estimation de mouvement, nous proposons une nouvelle méthode discriminative permettant la bonne reconnaissance des objets en mouvement.

I.2 Le mouvement dans une scène

Les producteurs de films, et des programmes de télévision, ont longtemps misé sur le fait que les êtres humains ont une anomalie dans leur système visuel. Quand un humain est confronté à une série rapide d'images fixes, l'esprit peut "remplir" les lacunes entre les frames et imagine qu'il voit un objet en mouvement continu. Par exemple, une série de flèches allumées lancées en succession sont perçus comme étant une seule flèche se déplaçant à travers l'espace. L'illusion de mouvement continu est appelé mouvement apparent pour le distinguer du mouvement «réel», qui est perçu quand un objet se déplace en continu à travers le champ visuel du spectateur. Lorsque un personnage dans un film bouge, il est en mouvement apparent, alors qu'une personne marchant sur l'estrade d'un théâtre est en mouvement réel. Dans ce derniers siècle, les cinéastes et les techniciens de ce domaine ont appris à créer beaucoup d'illusions convaincantes du mouvement, mais leur progression a été favorisé principalement par les mécanismes avec lesquels le système visuel de la rétine et cerveau perçoit le mouvement apparent.

I.2.2 Le mouvement réel et le mouvement apparent :

L'œil humain et les appareils d'acquisition visuelle (les caméras), observent les scènes réelles contenants des mouvements réels qui animent un espace en trois dimensions. D'apes le travail réalisé par Dufaux [DM 95], ce mouvement ne peut être directement mesurable. La projection du mouvement réel sur un plan en deux dimension est appelée le mouvement apparent ou le flux optique, ce mouvement se base sur la distribution de l'intensité des pixels. Ainsi on peut faire une segmentation basée mouvement on s'appuyant sur la distribution spatiale d'intensité lumineuse.

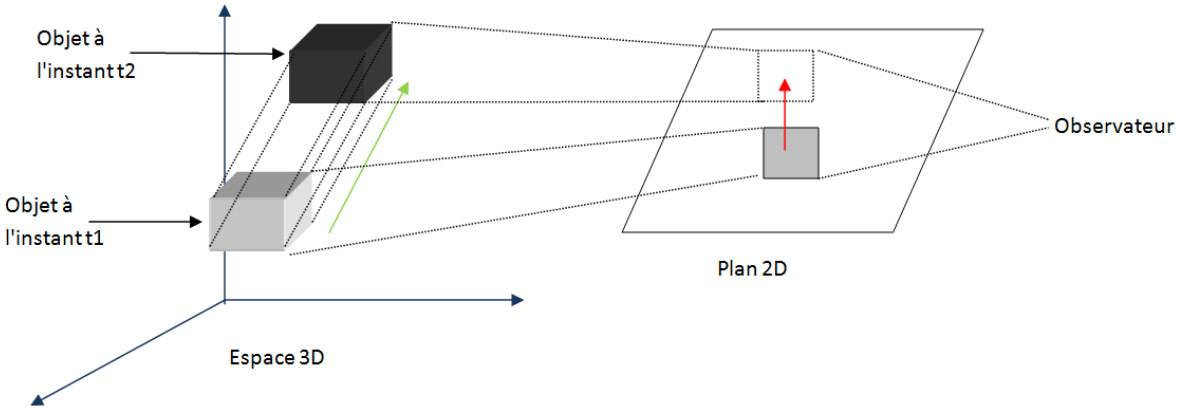


Figure.1.1 : Présentation des mouvements réel et apparent, dans un système optique de prise de vues.

Une fois on projette le mouvement réel d'une scène en trois dimension sur un plan en deux dimension (voir figure 1.1), on obtient se qu'on appel le mouvement projeté, ainsi la majorités des travaux s'appuient sur cette projection orthographique pour résoudre le problème de détection de mouvement[Hee 87].

On note par ${P^i}_t$ et ${P^i}_{t+1}$ l'ensemble des points représentants l'objet O^i à l'instant t et $t+1$. La projection perspective du mouvement de l'objet O^i est désigné par $\overline{P^t, P^t P_{t+1}}$. Concernant le vecteur de mouvement estimé $d(p) = \left(d_x(p), d_y(p)\right)$, il est extrait à partir des variations locales d'intensité lumineuse $I_t(x, y)$ entre les instants t et $t+ \Delta t$, qui représentent le champ de mouvement apparent. La variation temporelle du déplacement par unité de temps,$\left(v_x, v_y\right) = \left(d_x / d_t, d_y / d_t\right)$ représente le vecteur vitesse du mouvement $v = \left(v_x(p), v_y(p)\right)$.

I. 3 La segmentation du mouvement

la détection d'un objet en mouvement est la première étape importante pour un système d'analyse de mouvement utilisé dans le domaine de la surveillance, la compression vidéo... Une segmentation réussie d'un objet en mouvement et l'extraction de ce dernier de l'arrière-plan assure la bonne classification des objets mobiles, l'identification personnel, le suivi et l'analyse de l'activité. Une bonne détection rend ces étapes plus exactes. [Hu 04] a classé la détection de mouvement en trois grandes classes de méthodes: Différence d'image, flux optique, et la soustraction de fond. D'autre travaux qu'on va citer dans le prochain chapitre, utilise une approche discriminative au lieu d'une approche générative pour la détection de mouvement. Ils utilise les représentation parcimonieuse pour séparer les objets en mouvement du fond de l'image.

I.3.1. Détection basée sur la différence

La méthode la plus simple pour détecter le mouvement est proposée par [Jain]. Cette méthode consiste à faire la soustraction entre deux images successives. Cette approche n'est applicable que dans le cas où la caméra est fixe durant l'acquisition de la séquence. Pour obtenir le masque du mouvement $D(x,y)$, il est possible de soustraire à l'image courante $I_t(x,y)$ une image de référence $B(x,y)$ représentant uniquement la scène (sans la présence des objets en mouvement) .

$$D(x,y) = \begin{cases} 1, & si|I_t(x,y) - B(x,y)| \geq \tau \\ 0 & sinon \end{cases} \quad (1,1)$$

Où τ représente le seuil avec lequel on décide si le pixel fait partie de l'arrière plan ou d'un objet en mouvement. Les pixels dont l'intensité résultante est proche de zéro sont assimilés comme étant les pixels du fond. Le seuil τ doit tenir compte du bruit et des changements de luminosité. De plus, ce seuil n'est pas forcement global sur toute l'image. En effet, des variations d'intensité différentes sont générées par des différents mouvements ou d'autres phénomènes naturels (Le moment d'acquisition: matin ou soir; le climat: qui peut passer d'un état nuageux à un état dégagé). Pour remédier à ce problème et éliminer le bruit, on peut appliquer un filtrage du masque $D(x,y)$ obtenu par la formule (1,1); Ou d'adapter et mettre à jour l'arrière plan (l'image référence $B(x,y)$) selon les variations temporelles.

$$B_t(x,y) = (1-\alpha)B_{t-1}(x,y) + \alpha I_t(x,y) \quad (1,2)$$

Avec α représentant le taux d'apprentissage.

De nombreuses méthodes, utilisant deux ou trois images successives, permettent de décider si un pixel a bougé. Elles s'appuient généralement sur un test d'hypothèse. Une analyse des différentes méthodes de détection de changement permettant de détecter le mouvement est donnée dans [Radke 05].

I.3.2. Modélisation du fond

Pour la modélisation de fond, c'est une méthode qui consiste à construire un modèle d'arrière-plan, souvent multimodale, et le comparer à chaque image acquise. Les premières approches ont utilisé un modèle fixe comme une image d'arrière-plan prévu et la directement comparer avec les images entrantes. Nous avons adopté aussi ce procédé au niveau de l'implémentation de notre système (figure 1.2). Bien que cela fonctionne dans des situations idéales, il ne peut pas faire face à toutes variations d'éclairage et il s'avère ainsi limité. Une moyenne d'arrière-plan image peut être utilisée pour assurer un peu d'adaptabilité vis-à-vis les changements naturels, mais cela est encore limité. Récemment, des solutions multimodales [Butler 03, Stauffer 99] ont été proposées et qui sont en mesure de mieux faire face aux situations du monde réel et des problèmes tels que les alternances d'éclairage.

Figure.1.2 : Modèle de fond.

Le choix de l'espace couleur et le modèle de couleur est également important. Les espaces de couleurs tels que YCbCr et HSV facilitent la séparation de la couleur de l'intensité rendants ainsi quelques traitement plus facile tel que la détection de l'ombre.

Nous avons utilisé l'espace RVB pour comme proposé par Horprasert et al. [Horprasert 99] Dans leur travaux un modèle de couleur en fonction de l'espace RVB qui a peut permettre que la luminosité et la distorsion des couleurs soient bien mesurées. Une couleur donné dans l'espace RVB est représentée comme une ligne partant de l'origine à la couleur, et une ligne de chrominance. La distorsion de luminosité est une valeur scalaire qui relie la couleur observée à la ligne de chrominance, tandis que la distorsion des couleurs est la distance entre la couleur observée et la ligne de chrominance. L'utilisation d'un tel modèle de couleur permet une bonne séparation de la silhouette des objets de quelques information indésirable tel que l'ombre.

Dans Thongkamwitoon 04] les auteurs ont proposé une méthode de soustraction de fond adaptative basée sur le travail réalisé dans [Horprasert 99] qui peut varier le taux d'apprentissage pour les différentes parties de la scène. Un taux constant d'apprentissage peut entraîner des erreurs de deux façons:

1: Les informations constituants le fond peuvent être perdues dans les zones contenants de fortes activités lorsqu'un taux rapide d'apprentissage est utilisé (voir figure.1.3).

2. Les nouvelles informations de fond seront intégrés très lentement lorsqu'un taux lent d'apprentissage est utilisé(voir figure 1.4).

La figure 1.4 montre clairement que les données(pixels) représentants une personne sont confondues avec le fond. Tandis que pour la figure 1.3, le taux d'apprentissage rapide a fait que les pixels d'un objet (bras) son considéré comme un fond.

Pour surmonter ces problèmes, les auteurs [Horprasert 99] définissent dans leur approche un facteur d'intensité qui peut être utilisé en tant qu'une substitution de la vitesse d'apprentissage. cette valeur d'intensité est calculée par l'observation du changement des pixels d'une fenêtre à l'intérieur d'une image, et peut être utilisé pour compenser les différents niveaux de mouvement dans la scène.

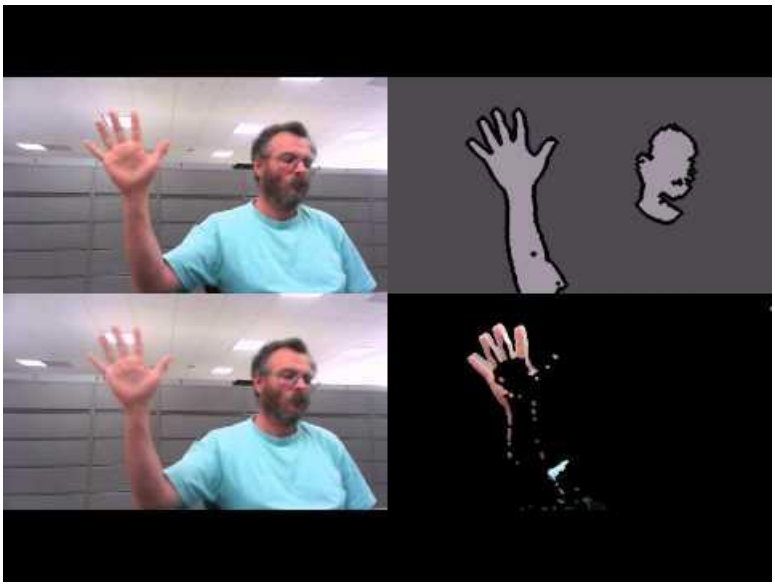


Figure 1.3: Modélisation du fond avec un taux d'apprentissage élevé

Figure 1.4: Modélisation du fond avec un taux d'apprentissage lent

Les filtres de Kalman sont généralement utilisés pour la modélisation du fond prédictives, il s'agit d'un model dynamique de prédiction basé sur les précédentes observation pour prédire la valeur d'un pixel [Karmann 90, Koller 94]. Dans les travaux réalisés par [Toyama 99], un algorithme nommé "wallflower" est proposé pour simplifier l'utilisation du filtre de Kalman appelée filtre de Weiner, ainsi les valeurs précédente servent à prédire la valeur d'un pixel au temps actuel.

Stauffer et Grimson [Stauffer 99] ont proposé un modèle d'arrière-plan multimodal utilisant un model gaussien (GMM) pour modéliser chaque pixel. Les pixels entrants sont comparés à la GMM pour déterminer dans quelle mesure elles satisfont l'arrière-plan (faisant partie ou non de l'arrière plan). Cette multimodale modélisation permis mieux l'utilisation d'un modèle d'apprentissage et d'adaptation à des changements dans l'arrière-plan. Cette approche basée sur le mélange de gaussiennes (MOGS) (ou un systèmes qui utilisent une approximation de ce [Butler 03]) est devenu populaire et plusieurs des améliorations et des variantes ont été proposées depuis son apparition.

Harville et al. [Harville 04] ont proposé un système basé sur les travaux de Stauffer et Grimson [Stauffer 99] et le travail de Gordon et al. [Gordon 99], afin de produire un système qui effectue une segmentation du l'avant plan et qui est basée sur la profondeur et la couleur des informations dans un cadre MOGS. Cet algorithme proposé dans ces travaux est capable de moduler le taux d'apprentissage conformément au taux de mouvement d'un pixel donné. les pixels qui sont les plus actifs, servent à un apprentissage plus lent à la préservation de l'arrière-plan, tandis que les pixels inactifs ont le taux d'apprentissage qui augmente car il est probable qu'ils représentent un nouvel objet d'arrière-plan.

Bowden et Kaew Trakulpong [Bowden 01] ont amélioré le système proposé par Stauffer et Grimson [Stauffer 99] par l'amélioration du taux d'apprentissage de sorte qu'il a converge plus vite sur un support de fond stable. Des changements ont été également proposés, qui permettent au système de mieux traiter l'existence de l'ombre. D'autres améliorations ont été proposées par Wang et Suter [Wang 05], qui a proposé d'ajouter une procédure de suppression de l'ombre et un modèle d'avant plan considéré comme une carte pour aider à la mise à jour de l'arrière-plan. L'approche MOG a également été appliquée à une situation de mouvement de caméra par Hayman et Eklundh [Hayman 03], et aussi adaptées par [Stauffer 99] à un système avec une caméra capable d'effectuer des mouvements horizontals et verticals (tel que peut être trouvé sur un robot, ou la vidéoconférence).

Un problème qui pourrait se poser avec ces approches, est que les objets qui ne font pas partie de l'arrière-plan, peuvent être incorporés dans le modèle du fond. Quand un objet s'arrête, d'abord il sera toujours détecté comme faisant partie de l'avant plan, cependant, après une période de temps suffisamment long l'objet présent dans la scène va être considéré comme une partie de l'arrière-plan. Si cela se produit, et l'objet commence alors à se déplacer de nouveau, il est possible que le mouvement sera incorrectement détecté à l'emplacement où

l'objet était. Bien que cela pourrait être en partie résolu en utilisant un lent taux d'apprentissage, ça sera alors une incidence sur d'autres aspects du système telles que la capacité d'apprentissage concernant les changements dans l'arrière-plan en raison de l'évolution des conditions de lumière (quelque chose qui est nécessaire dans une scène en plein air). Dans un système de suivi de d'objets, cette incapacité peut conduire à une augmentation du cas de l'occlusion, en rejetant ainsi le résolution du problème sur les autres étapes de segmentation ou de suivi et les étapes de prévisions. Tel est le cas de notre système, pour améliorer la précision de la détection, nous avons eu recours à l'utilisation des représentations parcimonieuses pour la reconnaissance des objets.

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

Figure 1.5: Stationnement temporaire de voiture. (a): frame 0 (b): frame 800. (c) frame 1900. (d) frame 2450.

Figure 1.5 montre plusieurs images d'une séquence où les voitures s'arrêtent temporairement, puis elles redémarrent. Dans ce cas, un véhicule est à l'arrêt dans la scène pour plus de 2000 images (80 secondes si les frames sont capturés à 25fps). Au même temps que la voiture est à l'arrêt, il y a des changements importants d'éclairage de la scènes (la différence entre la frame 1900 et 2450 de la figure 1.5). Dans une telle situation, si un taux d'apprentissage lent est utilisé pour détecter les objets en arrêt temporaire, le même taux d'apprentissage lent entraînerait une fausse détection de mouvement après le changement d'éclairage.

Dans [Javed 02] Les auteurs on ajouter plus d'informations de gradient au modèle MOG pour aider à surmonter le problème de changements d'éclairage car le gradient dans le fond demeure relativement stable en arrière-plan lors d'une variation d'illumination, alors que pour couleur ce n'est pas le même cas. Il est aussi possible de calculer la distribution du gradient à partir du modèle de l'arrière-plan en utilisant les différences gaussiennes qui représentent ces modèles de fond. Pour chaque pixel d'entrée, le gradient (amplitude et direction) peut être calculée et si elle correspond à une des distributions gradient, alors le pixel appartient à l'arrière-plan.

Zang [Zang 04] a proposé un mélange de modèle gaussiennes appelé "Pixel Map", qui combine une approche MOG avec les niveau de la région et l'image pour éliminer les zone incomplète de la forme des objets et aider à réduire le bruit.

I.4 L'estimation du mouvement

L'estimation du mouvement est un problème fondamental pour l'analyse de séquences d'images. Il consiste à mesurer la projection 2D dans le plan image d'un mouvement réel 3D, dû à la fois au mouvement des objets dans la scène et aux déplacements de la caméra.

Figure 1.6: Mouvement des pixels. (a) Image à l'instant *t.* (a) Image à l'instant *t+1.*

La figure 1.6 est un exemple qui illustre le changement de la position des pixels pendant le mouvement. Le principe de l'estimation de mouvement basé pixel ou région est de trouver le vecteur d de déplacement définit dans la section I.1.2.

I.4.1 Le flot optique

Le optique est flot optique est un procédé qui a comme but de déterminer le mouvement de chaque pixel d'un objet entre la scène acquis à cet instant et les images ultérieures.

$$I(x + dx, y = dy, t + 1) \cong I(x, y, t) \qquad (1,3)$$

Pour le calcul du flot optique on se base sur deux hypothèses:

1. Pour une région donnée de deux image consécutives, l'aspect ne changera pas en raison de l'éclairage (luminance constant).

2. Un pixel présent dans un image sera toujours présent dans l'image suivante (pas de discontinuités spatiales).

Ces deux conditions doivent être respectées, sinon des erreurs d'estimations pour subsister.

La deuxième hypothèse n'est pas respectée dans le cas d'occultations, de transparences, de réflexions spéculaires et plus généralement de variations brutales de luminosité. L'invariance temporelle entre deux instants peut être caractérisée par une formulation différentielle linéaire, aussi appelée équation de contrainte du mouvement apparent (ECMA) :

$$\frac{dI}{dt}(x,y) = \frac{dI}{dt}(x,y) + \nabla I(x,y).(dx,dy) = 0 \qquad (1,4)$$

Pour quelques application de la vision par ordinateur comme par exemple la vidéo surveillance, l'utilisation du flot optique peut avoir des limites et des inconvénients car il détecte tous sorte de mouvement. Des techniques de modélisation de fond peuvent être utilisées pour filtrer les mouvements répétitifs tels que le mouvement des arbres qui se balancent à cause du vent, qui sera détecté par le flot optique. Cependant, plusieurs traitements supplémentaires peuvent être nécessaire pour filtrer l'image afin de déterminer le mouvement qui est causé par les objets cibles de la scène.

I.4.1.1 Le problème d'ouverture

D'après l'équation (1.4), seule la composante de la vitesse parallèle au gradient spatial d'intensité, appelée vitesse normale, est directement calculable. Ce problème classique et commun aux différentes approches d'estimation du mouvement est désigné sous le nom de "problème de l'ouverture".

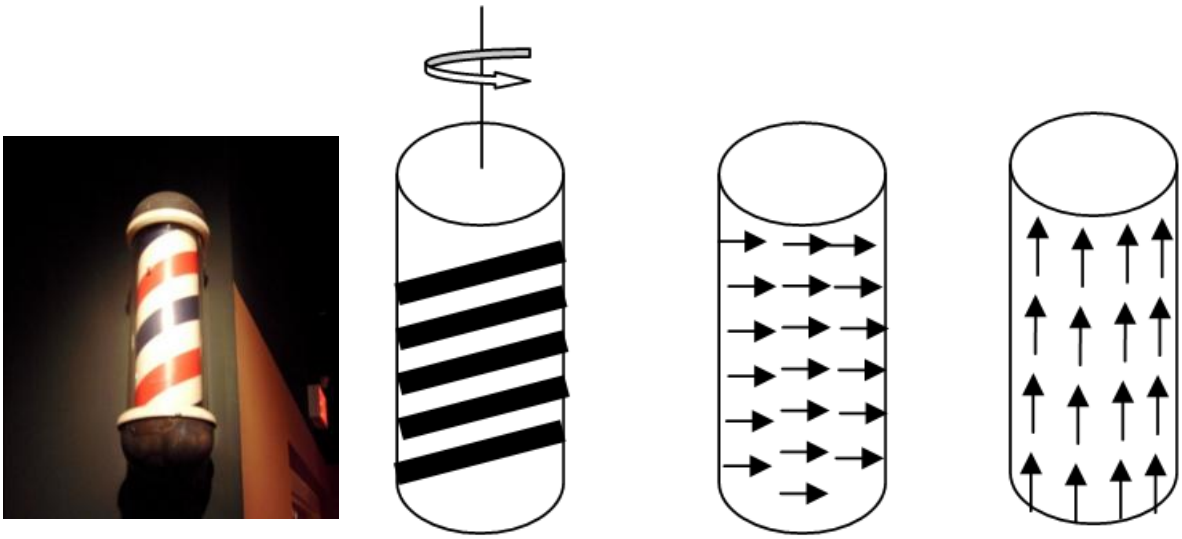


Figure 1.7: Problème d'ouverture. (a) Forme cylindrique hachurée et en rotation sur l'axe Z. (b) Une autre représentation de cette forme cylindrique (c) Le flot optique supposé être détecté. (d) Le flot optique supposé engendré.

Le mouvement représenté par la figure 1.6 illustre bien le problème d'ouverture. Comme c'est le cas pour l'être humain de croire que les traits du cylindre bouge vers le haut. Le flot optique souffre aussi de ce problème, est le vecteur de mouvement obtenus sont dirigés vers le haut (Figure 1.7.d).

Une autre illustration peut être aussi donnée avec la figure 1.8. Dans quelque cas on ne peut déterminer le nouvel emplacement d'un pixel d'une image à une autre en raison que plusieurs autres pixels on la même intensité. Il s'agit aussi du problème d'ouverture car l'équation (1.3) admet dans ce cas plusieurs solutions.

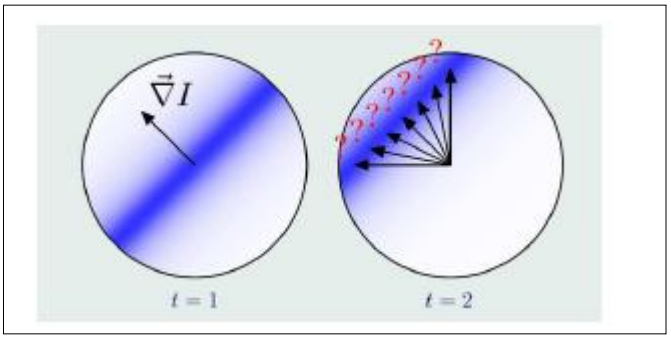


Figure 1.8 Le problème d'ouverture : pour une position donnée x l'équation du flot n'admet pas de solution unique (non-unicité) car elle a deux inconnues, les composantes de v. La direction du vecteur vitesse est donc a priori inconnue.

La plupart des techniques de flot optique sont des méthodes basée sur le gradient (Horn et Schunck [Horn 81], Lucas et Kanade [Lucas 81]), ou sur la correspondance de blocs ([Bergen 90], [Burt 89]) qu'on discutera par la suite. Les méthodes à base de gradient sont plus préférées en raison de leur rapidité et de leur performance. Les méthodes basées sur le gradient analysent le changement de l'intensité et le gradient (en utilisant la dérivé partielle spatiale et la dérivé partielle temporelles) pour déterminer le flot optique. La figure 1.9 est un exemple du vecteur mouvement obtenu à partir du calcul du gradient et de son orientation. . Les méthodes basées sur la correspondance de blocs reposent sur la détermination de la correspondance entre les deux images. Cela implique généralement l'appariement d'un blocs d'une image à des blocs de l'autre image, afin de déterminer dans quelle mesure cette région c'est déplacée.

Figure 1.9 Exemple du calcul du flot optique

I.4.2 La mise en correspondance

La méthode la plus populaire dans le domaine d'estimation de mouvement est l'algorithme du "Block Matching" (BMA). Cet algorithme est adopté par plusieurs standard de compression vidéo tel que: H.261/H.263 et MPEG-1/2/4.

Estimer le mouvement entre deux images revient à faire correspondre les éléments de la première avec ceux de la deuxième. On pourra les lier ainsi par un champ de vecteurs.

Le principe pour le BMA consiste à découper la première image en zones plus ou moins grandes. Pour chacun de ces blocs (appelées aussi zones d'intérêt ou encore fenêtres

d'interrogation), on parcourt la seconde image (la totalité ou une partie) à la recherche de la zone qui lui ressemble le plus. Pour cela, on résout le problème d'optimisation.

$$\underset{\delta x,\delta y}{argmax} \int_{v(x,y)} I(x'+\delta x, y'+\delta y, t+1) I(x', y', t) dx' dy' \qquad (1,4)$$

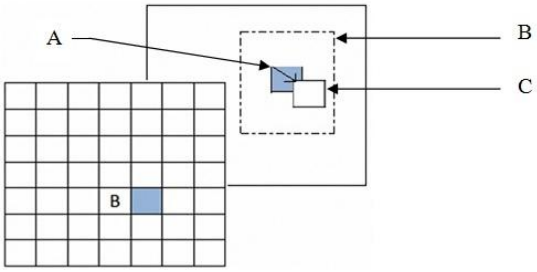


Figure.1.10 Le BMA

Il existe plusieurs méthodes de recherche de correspondance; en compte parmi elle: la recherche exhaustive, parcours en spirale, en dimons. On présentera par la suite chacune de ces méthodes qui diffèrent en terme de qualité du résultat et en terme de complexité.

La recherche exhaustive: appelée "Full search" (FS), cette méthode est la plus simple parmi les méthodes du BMA, elle garantie de bon résultats pour la mise en correspondance des bloques. Le FS atteint des performance optimale en analysant tout les points possible dans la zone de recherche appartenant à l'image référence. Par contre, cet algorithme est très coûteux point de vue temps d'exécution, et l'utiliser pour un système à temps réel devient une tache difficile.

I.4.2.1 Modèle de mouvement

Le mode de translation ou le modèle constant de mouvement est le plus utilisé, qui suppose que tous les pixels du bloc effectuent le même déplacement. (Figure 1.11)

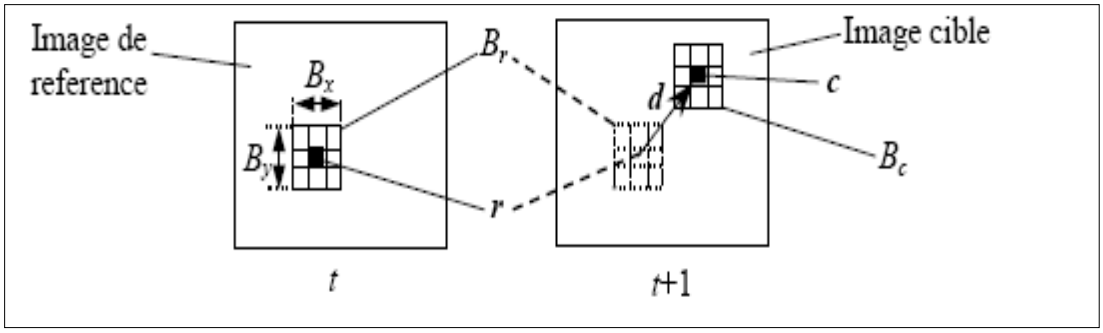


Figure 1.11 : Modèle de translation du mouvement.

Pour chaque bloc de pixels dans une image de référence

Ainsi, un bloc *Br* de pixels et de dimensions *Bx*×*By* de l'image de référence, de centré sur le pixel (***r***) de coordonnées (*x*,*y*) dans l'image de référence à l'instant *t*, sera mis en correspondance avec le bloc *Bc* centré en pixel ***c***, dans l'image cible à l'instant *t*+1, ce qui s'écrit:

$$Br\,(x, y, t) \;=\; Bc\,(x \;+\; dx\,, y \;+\; d\,y\,, t \;+ 1) \qquad (1.5)$$

I.4.3 Méthodes non paramétriques

On utilise les méthodes non paramétrique pour la régularisation des mouvement complexes. On cite dans cette partie les approches principale:

I.4.3.1 Les méthodes différentielles

Le principe des méthodes différentielles et de calculer le gradient spatio-temporels de l'intensité de chaque pixel [Horn 81]. Pour les images monochromatiques, ces méthodes font l'hypothèse que le vecteur déplacement varie lentement au voisinage du pixel en respectant une contrainte de lissage spatio-temporel. Dans le cas des images couleurs, cette contrainte peut être imposée séparément pour chaque couleur, ce qui peut contraindre le vecteur déplacement dans 3 directions. On peut ajouter une contrainte supplémentaire afin d'avoir de bon résultats.

Lorsqu'on est confronté au problème d'occlusion, une contrainte globale est ainsi appliquée.

I.4.3.2 Les méthodes récursives

La prédiction du vecteur déplacement est corrigé en récursivité, d'où l'appellation de ces méthodes(récursives). La prédiction est représentée comme une combinaison linéaire ou un vecteur de déplacement d'un pixel voisin dans un voisinage de pixel courant. On minimise le gradient de l'image de différence entre les images déplacées DFD pour corriger la prédiction.

Le pas de la prédiction est considéré, en général, comme une contrainte de lissage. Une extension de cette méthode dans le cas des méthodes de mise en correspondance de blocs, a comme résultat les méthodes d'estimation de type Wiener [Kon 00].

I.4.3.3 Les méthodes statistiques

Parmi les méthodes statistiques, les méthodes Markoviennes ou Bayesiennes sont les plus répandues. Elles utilisent pour estimer le champ de déplacement des contraintes probabilistes de lissage, en général, sous la forme d'un champ aléatoire (éventuellement de Gibbs). Le principal désavantage de ces méthodes consiste en la grande quantité de calculs nécessaires pour estimer le champ de mouvement [Geman 84].

1.5 Les méthodes discriminatives pour la détection de mouvement

Les travaux les plus récents de détection de mouvement (mouvement de la main, visage, personnes ..) se base sur des approches discriminatives qui séparent l'avant plan (des objets) du fond. ces objets détectés sont supposés des régions de l'image en mouvement. Ainsi une classification est établi; et par la suite un système de suivi peut être utilisé pour estimer le mouvement de chaque objet.

De nombreuses approches de classification, de méthodes basées sur les caractéristiques et des modèles de déformation ont été utilisées pour améliorer la détection d'objets. Les méthodes de classification les plus utilisées incluent des divers systèmes [Dollar 09, Wu.B 05], système linéaire SVM [Dalal 05], intersection de noyau histogramme SVM [Maji 08], latente SVM [Felzenszwalb 10], multiple noyau SVM [Vedaldi 09] et le SVM à structures [Zhu 10].

Plusieurs approches représentent des objets à l'aide de fonctions de bas niveau. Par exemple, dans [Amit 99, Roth 02], les parties de l'objet sont des regroupements locaux communs de

fonctions primitives tels que des fragments de bord, tandis que dans [Viola 01], les caractéristiques du rectangle contenant l'objet sont utilisées pour capturer la présence des arêtes, traits et autres structures simples. L'utilisation d'une représentation partielle peut être améliorée en considérant, les pièces de niveau supérieur qui sont très riches en contenu d'informations et plus spécifiques à la classe d'objets d'intérêt. Une telle approche a été utilisée dans [Mohan 01], dans lequel des classificateurs distincts sont utilisés pour détecter la tête, les bras et les jambes de personnes dans une image, et un classificateur final est ensuite utilisé pour décider si oui ou non une personne est présente. Cependant, le travail dans [Mohan 01] exige que les parties d'objet à définir et séparés pour la formation des classificateurs de pièces individuelles soit manuelle. Afin de construire un système qui est facilement extensible pour faire face à différents objets, il est important que la procédure de sélection de partie doit être automatisé. Autre méthode pour sélectionner automatiquement l'information des parties riches repose sur une technique décrite dans [Weber 00], dans lequel les points d'intérêt sont utilisés pour recueillir des parties distinctes. Dans [Weber 00], un modèle probabiliste générative est enrichi au fur et mesure. Notre méthode, d'autre part, n'a pas besoin de supposer un modèle probabiliste; nous recueillons simplement des parties qui pourraient être d'intérêt et ensuite apprendre directement un classificateur sur eux. En outre, le modèle appris dans [Weber 00] repose sur un très petit nombre de parties fixes, ce qui en fait potentiellement sensible à de grandes variations entre les images. En apprenant sur un grand espace de caractéristiques, nous sommes en mesure d'apprendre un modèle plus expressive qui est robuste à de telles variations.

I.6 Détection et suivi d'objets

Comme mentionné au niveau de l'introduction, nous portons notre intérêt au suivi d'objet mobile dans une séquence d'image. Nous allons décrire ce principe ainsi et nous donnons un bref état de l'art de ce domaine.

Le processus de suivi d'objet peut être abordée de deux façons:

1. Dans chaque image, détecter tous les objets et les regrouper dans la liste des objets à partir de la dernière image;

2. détecter un objet et extraire une ou plusieurs caractéristiques pour le décrire, puis suivre cet objet en utilisant les caractéristiques extraites.

La première approche est l'approche la plus commune de suivi à l'objet. Pour la seconde approche on cite quelque techniques tel que l'algorithme de changement de vitesse moyenne [Fukunaga 90] et ses dérivés (par exemple, Camshift, continue Adaptive Shift moyenne, [Bradski 98]), et de filtres à particules. Ces systèmes reposent souvent sur une entrée externe pour initialiser le processus de suivi, comme un procédé de détection en continu qui permet modes de découverte automatique de nouveaux modes, il nécessite de comparer les détections afin de déterminer quels sont les objets déjà en cours de suivi. Pour la tâche de surveillance automatique, il est souhaitable de pouvoir automatiquement découvrir des objets quand ils entrent dans la scène, comme en s'appuyant sur un opérateur humain pour attribuer aux objets entrants un nom ou un drapeau.

I.6.1 Détection d'objets

Afin de suivre un objet, le système doit être capable de le détecter et de pouvoir bien extraire les caractéristiques qui peuvent être observés et combinées d'une image à une autre. Ces caractéristiques peuvent être telle que des caractéristiques basées sur la position et la distance, ou basées ou des fonctions plus complexes de couleurs et de textures. Les paramètres d'acquisition (de couleur ou niveau de gris, résolution de l'image, champ de vision de la caméra) et de l'environnement (intérieur / extérieur, jour / nuit), dans lequel le système est destiné à fonctionner est susceptible de jouer un rôle important dans la détermination du type de caractéristiques utilisé . De nombreux systèmes utilisent la détection de mouvement pour détecter des objets et le suivi. Haritaoglu et al. [Haritaoglu 99] détecte les personnes en plaçant des blobs de mouvement et calcule l'histogramme verticale de la silhouette (têtes se situera à un local maximal), et en combinant cette informations avec les résultats de sommets convexes pour déterminer où potentiellement la têtes se situait dans la région. Cette approche peut être utilisée efficacement pour des groupes de segments de personnes. Fuentes et Velastin [Fuentes 03] réalisent la détection de mouvement en utilisant la luminance contraste et les blobs formés, les objets (personnes) détectés sont caractérisés par un cadre de sélection, centre de gravité, la largeur et la hauteur. Un blobs est un groupe de régions connectées, en fonction des contraintes spatiales. En tant que tel, un blob détecté peut être constitué par une (8 ou 4) régions uniques, grandes, qui sont reliées; ou un groupe de plusieurs composants connectés petits situés près de l'autre. Plutôt que de suivre simplement les blobs, Zhao et Nevatia [Zhao 04] ont proposé un système qui utilise un modèle de forme ellipsoïde pour

localiser les personnes du secteur de l'image en mouvement. Le système a été installé de telle sorte que la caméra a été déployée à quelques mètres au-dessus du sol, regardant vers le bas pour aider à surmonter les problèmes d'occlusion qui se produisent avec des caméras au niveau du sol.

Les personnes sont détectées par un processus itératif. Les deux étapes suivantes sont répétées jusqu'à ce qu'il n'y est plus de personnes détectées dans la scène.

1. Localiser toutes les têtes et ajuster un modèle de personne ellipsoïde à la tête. S'il y a mouvement suffisante au sein de l'ellipse, la personne est acceptée et leur mouvement est retiré.

2. Effectuer une analyse de l'ombre géométrique pour éliminer des zones d'ombre appartenant à des personnes détectées (en utilisant la date, le lieu et l'heure de la journée pour déterminer la position des soleils, et l'orientation de toutes les ombres).

Kang et al. [Kang 03] a également appliqué la détection de la tête après la segmentation du fond pour localiser les personnes, qui sont caractérisés par des boîtes de délimitation. En outre, les résultats de la détection de la tête de la trame précédente sont utilisés dans la trame suivante à travers une boucle de rétroaction pour faciliter le processus. La détection du mouvement à base d' approches similaires ont été utilisées pour détecter les véhicules. Koller et al. [Koller 94] ont utilisé un modèle d'adaptation d'arrière-plan pour localiser des véhicules sur la route. La forme de l'objet est d'abord tirée à la fois par le gradient de l'image et le masque de mouvement. La silhouette est exprimé sous la forme d'un polygone convexe qui entoure l'objet, lissé par l'application de paramètres de spline cubique des points.

Un problème qui peut se produire lors de l'utilisation des résultats de la segmentation basée mouvement qui est la base pour la détection d'objet est que le mouvement parasite peut être inclus dans le cadre d'un objet détecté. Pour aider à surmonter cela, Lei et Xu [Lei 05] ont proposé un système de suivi qui utilise une variante de la métrique de distorsion de luminosité [Horprasert 99] pour éliminer les ombres et les autres bruits. Ce filtrage (élimination de l'ombre / lumière) est réalisée à deux niveaux de seuil (serré et lâche). Les blobs qui ne sont pas liés au cadre de l'objet en sortie du seuillage sont regroupés en fonction de la connectivité du seuil bas le plus bas. Cela permet au regroupement de blobs en vrac tout en veillant à ce que plus de le mouvement parasite est supprimé.

les systèmes de suivi qui utilisent des images en couleur en entrée comme Matsumura et al. [Matsumura 02] et Wang et al. [Wang 02] utilisent la détection de la peau pour localiser les personnes au sein de l'image. Ces régions de la peau sont suivies, et un modèle est construit à partir de segments extraits de la peau. Ce modèle de segment de la peau peut alors être utilisé pour faire correspondre les candidats dans des trames futures. Wang et al. [Wang 02] et Matsumura et al. [Matsumura 02] ont appliqué la détection du mouvement pour simplifier la détection de la peau en supprimant les régions sans intérêt. Matsumura et al. [Matsumura 02] ont recherché la couleur de la peau dans des cadres, et ont suivies les grandes régions de peau. Ces systèmes, cependant, dépendent de la capacité à détecter le visage ou d'autres régions de la peau pour pouvoir suivre et cela peut ne pas être toujours possible.

D'autres systèmes utilisent une approche de modèle d'apprentissage plutôt que de compter sur la détection de mouvement. des approches basées sur le mouvement sont finalement dépendant de la performance de la détection de mouvement, avec une détection de mouvement pas très fiable le système est susceptibles de conduire à une mauvais suivi d'objet. Une approche basée sur un modèle d'apprentissage peut surmonter cette limitation.

Rigoll et al. [Rigoll 00] ont utilisé un pseudo 2D HMM (P2D HMM) pour suivre les personnes. Ceci pour éviter d'utiliser la détection de mouvement, et ainsi éviter les problèmes tels que (les voitures / autres personnes / arbres provoquant un mouvement supplémentaire) qui se traduit par de faux objets détectés dans la scène ou des pistes (trajectoires) valables perdues. Un P2DHMM formés sur plus de 600 personnes a été utilisé pour localiser des personnes dans une frame. Le P2DHMM utilise le suivi des objets centroïde, la vitesse, la hauteur de la boîte et la largeur comme entée du filtre de Kalman [Welch 95]. Le filtre de Kalman alimente la prédiction suivante à l'HMM pour faciliter le suivi, et pour que le suivi basé sur le HMM progresse, il adapte son modèle pour mieux répondre à l'objet qu'il utilise actuellement. Le principal avantage d'un tel système est qu'il n'y a pas de détection de mouvement, ce qui signifie que la caméra peut zoomer sans qu'il en résulte toute complexité supplémentaire étant ajoutée au système.

I.6.2 Reconnaissance et suivi d'objets

Une fois que les objets ont été détectés dans une image, il est nécessaire de faire correspondre ces objets à ceux qui ont été détectés dans l'image précédente. Zhao et Nevatia [Zhao 04] trouvent les objets détectés d'une manière itérative. Les pistes sont personnalisées une par une (chaque piste par rapport à tous les objets situés) dans l'ordre de leur profondeur dans la scène (déterminée par la position dans l'étalonnage de l'image et la caméra). Fuentes et Velastin [Fuentes 01] reconnaissent et suivent les objets détectés en utilisant une matrice et selon deux façons d'utilisation de l'algorithme d'appariement (correspondant à l'avant et àl'arrière). Afin d'effectuer cette correspondance, une certaine forme de fonction doit être utilisée pour la comparaison.

Les caractéristiques utilisées pour correspondre les trajectoires varient considérablement, et le type de fonction utilisées partiellement dépendant de la configuration du système et le type d'entrée du système (à savoir :niveaux de gris ou des images couleur). Les types de caractéristiques, qu'on peut utiliser globalement sont regroupés comme suit:

1. Les caractéristiques géométriques (par exemple la position de l'objet, position de la boîte, sa taille [Lei 05, Matsumura 02]) peuvent être extraites directement à partir des résultats de la détection d'objets sans traitement ultérieur de l'image. La fiabilité des caractéristiques dépend directement sur les performances de détection d'objet, mais les caractéristiques sont très rapides à extraire et comparer. En raison de leur simplicité, les caractéristiques géométriques sont souvent utilisé en combinaison avec des fonctions plus complexes [Lei 05, Matsumura 02].
2. Les caractéristiques basées sur le contour (comme la silhouettes [Haritaoglu 98, [Haritaoglu 98]) peut être extrait à partir d'un masque de mouvement, ou une image masque similaire (un masque pourrait être extrait à l'aide des techniques de segmentation couleur). La performance des caractéristiques de contour est en fonction de la précision du masque, les erreurs de segmentation seront conduisent à une réduction de performance.
3. Caractéristiques Couleur / Texture (histogrammes [Lu 01, NP 02], les modèles d'apparence [Haritaoglu 98, Hu 2 04, Kang 03, Rao 00]) peuvent être extraites en utilisant une combinaison de l'objet résultant de la détection des images d'entrée et les images de masquage. Ces caractéristiques sont plus robuste détection et la segmentation des erreurs, mais ces caractéristiques exigent plus de calculs.

I.7 Conclusion

Dans ce chapitre, nous avons présenté les méthodes de base réalisée au niveau de l'état de l'art. Ces méthodes de détection de mouvement varient selon le cas d'acquisition; et chacune d'elle a ses avantages et ses inconvénients. Dans notre travail, nous nous somme basés sur la soustraction pour la détection de mouvement. Pour le suivi et la reconnaissance nous avons combiné deux caractéristique (citées dans la section I.6.2) qui sont les caractéristique couleurs et position. Pour pouvoir bien reconnaitre les personnes et augmenter la performance de notre système, nous avons utilisé un système de classification. D'autre approche des l'état de l'art utilisent des système d'apprentissage comme cité dans la section I.6.1.

Chapitre II

Les architectures parallèles

II.1 Les architectures parallèles

Depuis l'apparition des ordinateurs, la puissance de calcul n'a fait que croître afin de résoudre des problèmes de complexités croissantes. Le calcul scientifique a connu une évolution rapide. Il est largement utilisé et se développe dans tous les domaines, tels que, la prévention météo, la simulation scientifique, mécanique, aérodynamique, électrique, biologique [Thuan 11, Chariot 09], etc...La puissance de calcul repose sur deux grandes notions:

– la latence de traitement ;

– le débit de traitement.

La première notion, la latence de traitement, représente le temps nécessaire pour l'exécution d'un traitement tandis que le débit de traitement représente le nombre de traitements exécutables par unité de temps. Ces deux notions peuvent être indépendantes. L'augmentation de la puissance de calcul consiste dans la réduction de la latence de traitement ou dans l'augmentation du débit de traitement. La réduction de la latence de traitement est plus difficile à obtenir que l'augmentation du débit de traitement. De plus, la puissance de calcul dépend aussi de la capacité et de l'organisation de la mémoire d'un ordinateur. Dans le modèle séquentiel, un programme est exécuté par un unique processus. Ce processus a accès à la mémoire du processeur. Plusieurs applications, telles que la simulation, la modélisation et l'optimisation numérique, requièrent des ensembles de données (météo, SIG, marketing, etc) dont la taille est supérieure à la capacité d'adressage d'un ordinateur séquentiel.

De plus, les exigences du temps de calcul dans ces applications sont importantes. La solution unique, afin de surmonter ces problèmes, est le parallélisme. Certaines organisations d'architectures parallèles permettent donc d'adresser plus de mémoire que des architectures séquentielles.

Historiquement, l'évolution des architectures parallèles a commencé à partir de l'architecture séquentielle de Von Neumann qui offre un flux de contrôle unique dont les instructions exécutent séquentiellement les opérations sur des opérandes scalaires. Une révolution informatique s'est produite lorsque le calcul scientifique basé sur les machines de type "mainframes" a été remplacé par celui basé sur le réseaux. Dans les années 1970s, l'introduction des systèmes vectoriels a marqué le début des supercalculateurs modernes. De

nombreux ordinateurs parallèles commerciaux sont ensuite apparus dans les années 1980s et 1990s. Ils peuvent être généralement classés en deux catégories principales : les systèmes à mémoire partagée et les systèmes à mémoire distribuée. Au début des années 1990s, alors que les systèmes multiprocesseurs vectoriels ont atteint leur plus large diffusion, une nouvelle génération de machines massivement parallèles a été introduite sur le marché.

Ces systèmes se sont avérés autant ou plus performants que les systèmes vectoriels. Le nombre de processeurs peut atteindre une centaine voir un millier de processeurs

dans un système massivement parallèle. Les calculs scientifiques, liées à ces architectures de machines, deviennent de plus en plus ambitieux. Mais les algorithmes, les paradigmes et les méthodes doivent évoluer pour pleinement tirer partie des points forts de ces architectures.

Nous présentons ici un bref panorama des architectures parallèles, de leurs contraintes et des moyens de les programmer avant de conclure sur nos choix.

II.1.1 Classification des Architectures parallèles selon Flynn

Une architecture parallèle est essentiellement un ensemble de processeurs qui coopèrent et communiquent. Les premières architectures parallèles furent des réseaux d'ordinateurs et des machines vectorielles faiblement parallèles (IBM 360-90 vectoriel, IRIS 80 triprocesseurs, CRAY 1 vectoriel, etc).

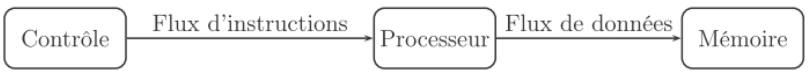


Figure 2.1 Architecture SISD

La classification la plus utilisée est historique, elle est connue sous le nom de classification de Flynn [Flynn 72]. Elle classe les architectures en fonction des relations entre les unités de traitement et les unités de contrôle. Les unités de traitement calculent chacune un flux de données (Data Stream en anglais, ou DS). Les unités de contrôle déroulent chacune un programme qui consiste en un flux d'instructions (Instruction Stream en anglais, ou IS). Chacun de ces types peut avoir un seul des deux états possibles : mono ou multiples. On peut trouver cette classification dans la Table II.1.

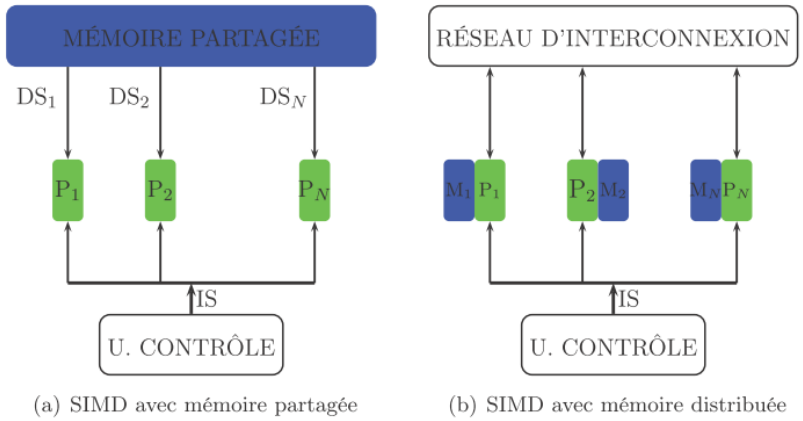
Table II.1 – Classification des architectures parallèles selon Flynn

Flux	mono flux d'instructions	multiples flux d'instructions
mono flux de données	SISD	MISD
multiples flux de données	SIMD	MIMD

L'architecture SISD (Single Instruction Single Data en anglais) est une architecture séquentielle. Elle correspond au modèle classique de Von Neuman illustrée dans Figure 1.2. Le traitement n'est effectué qu'avec un seul flux d'instructions sur un seul flux de données. Cette architecture peut être combinée avec la technique du pipeline afin de permettre l'exécution de plusieurs instructions. L'idée consiste à subdiviser les instructions en une série d'opérations plus fines et chacune des opérations peut être exécutée dans une seul phase de pipeline à moment donné.

La deuxième architecture (voir Figure 2.2) s'appelle SIMD (Single Instruction Multiple Data en anglais). Une seule unité de contrôle gère le séquencement du programme pour plusieurs unités de traitement. En pratique, les unités de traitement fonctionnent de manière synchrone et reçoivent les mêmes instructions en même temps. Comme chaque unité de traitement calcule sur un flux de données

différentes, la même opération est appliquée à plusieurs données simultanément. Cette architecture est ainsi capable de traiter plusieurs données à la fois. On classe dans cette catégorie également les machines vectorielles, les réseaux cellulaires et les réseaux systoliques.



(a) SIMD avec mémoire partagée (b) SIMD avec mémoire distribuée

Figure 2.2: Architecture SIMD (Single Instruction Multiple Data)

L'architecture MISD (Multiple Instruction Single Data en anglais) peut exécuter plusieurs instructions en même temps sur un seul flux de données (voir Figure 2.3). Le mode pipeline d'exploitation du parallélisme de contrôle correspond assez bien à cette classe. Le pipeline du type MISD correspond à un enchaînement de macro blocs, chacun ayant sa propre unité centrale.

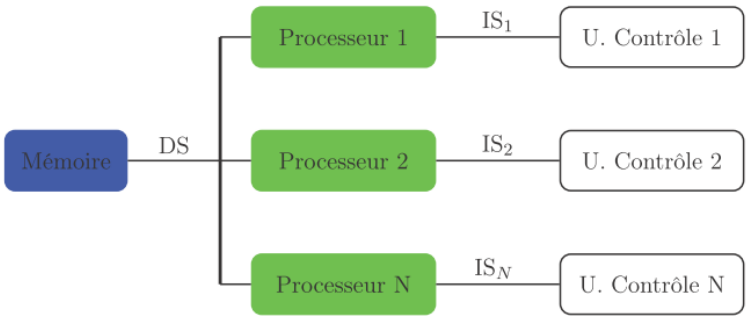


Figure 2.3 – Architecture MISD (Multiple Instruction Single Data)

L'architecture MIMD (Multiple Instruction Multiple Data) est l'architecture parallèle la plus utilisée. Elle est constituée de plusieurs processeurs et plusieurs modules de mémoires reliés entre eux par le réseau d'interconnexion. Cette architecture contient plusieurs unités de contrôle et chacune peut gérer une ou plusieurs unités de traitement. À tout moment, elle permet à différents processeurs d'exécuter des instructions différentes sur différentes données.
Dans l'architecture de MIMD, le modèle de programmation SPMD (voir Figure 2.5) (Single Program Multiple Data en anglais) est le plus commun pour la programmation. Un seul programme doit être écrit et sera exécuté par tous les processeurs.
Pendant l'exécution, les processeurs travaillent sur des données différentes et il faut ainsi écrire les parties correspondantes aux émetteurs et celles correspondantes aux récepteurs des communications au sein d'un même programme.

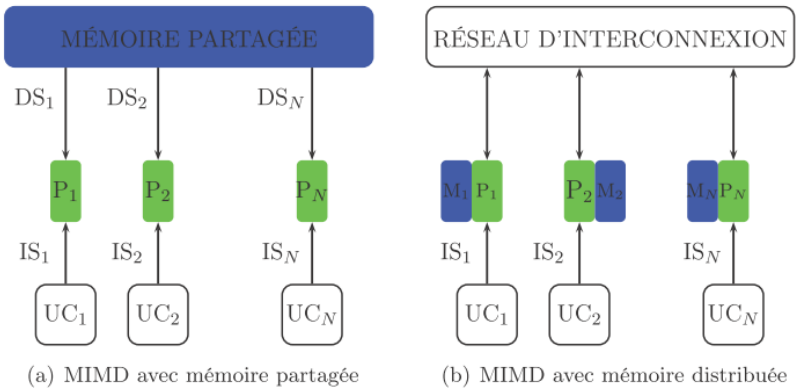


Figure 2.4 – Architecture MIMD (Multiple Instruction Multiple Data)

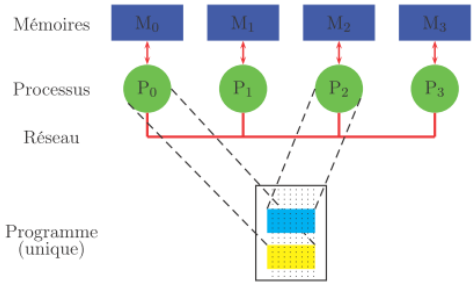


Figure 2.5 – Modèle de programmation de SPMD

II.1.2 Classification des Architectures parallèles selon la mémoire

La taxonomie de Flynn ne caractérise pas l'organisation de la mémoire vis-à-vis des processeurs. Or la réalisation de mémoires rapides de grandes tailles est nécessaire pour les architectures parallèles dédiées à la haute performance. Il existe deux grands types de machines parallèles : les machines à mémoire partagée et les machines à mémoire distribuée.

II.1.2.1 Machines à mémoire partagée

Les machines à mémoire partagée permettent de réaliser le parallélisme de données et de contrôle. Le programmeur n'a pas besoin de spécifier la distribution des données sur chaque processeur. Il définit seulement la partie du programme qui doit être parallélisée (directives) et doit gérer les synchronisations.

Dans une machine à mémoire partagée, les données d'une application sont toutes placées dans la mémoire commune, un processus p placé sur un processeur P a ainsi la possibilité physique d'accéder aux données d'un autre processus q placé sur un autre processeur Q. Cette

propriété est très utile car elle permet aux processus d'échanger des informations à travers des variables partagées. Les processeurs communiquent entre eux via la mémoire globale et par conséquent le temps de communication entre deux processeurs différents ne dépend pas de la position du processeur dans le réseau. Chaque processeur peut avoir une mémoire locale suffisamment grande pour le traitement local et le stockage temporaire. La cohérence entre ces mémoires locales devra être gérée par le hardware, le software et parfois l'utilisateur. En se basant sur le temps d'accès à la mémoire, il existe deux classes principales d'architectures parallèles à mémoire partagée. La première classe, telle que les machines dual/quad core x86 (PowerPC, SPARC, etc), se base sur l'organisation de type UMA (Uniform Memory Access en anglais). Cette architecture est constituée de plusieurs processeurs identiques connectés à une unique mémoire physique (de l'ordre de 2 à 4 processeurs). Tous les processeurs accèdent à n'importe quel point de la mémoire en un temps uniforme. Cette architecture est implémentée pour les machines SMP (Symmetric Multi processors en anglais). La deuxième concerne l'organisation de type NUMA (Non-Uniform Memory Access en anglais) dans laquelle l'architecture parallèle est constituée de plusieurs processeurs connectés à plusieurs mémoires distinctes s'appelant bancs de mémoires. Ces processeurs sont reliées entre elles par des mécanismes matériels (jusqu'à 2048 processeurs). Le temps d'accès à la mémoire globale n'est pas uniforme, il dépend du placement des données dans celle-ci.

Le point fort des machines parallèles à mémoire partagée est la simplification au niveau de programmation et le partage des données entre les tâches est aussi rapide. La difficulté principale de cette architecture est l'accès à la mémoire (conflits d'accès à la mémoire) qui limite le nombre processeurs. Le programmeur est responsable de la validité des synchronisations. À partir de quelques dizaines de processeurs, les performances commencent à se dégrader.

II.1.2.2 Machines à mémoire distribuée

Dans la machine à mémoire distribuée, chaque processeur possède sa propre mémoire locale et il n'a pas accès à celle des autres. Il exécute ses instructions sur ses données. L'accès à la mémoire du processeur voisin se fait par l'échange de messages à travers le réseau. Pour cela, les algorithmes utilisés sont implémentés de telle manière à minimiser les coûts de communications.

Ces communications entre les processeurs sont réalisées par l'appel à des fonctions de bibliothèque standard : PVM (Parallel Virtual Machine) ou MPI (Message Passing Interface)

via un réseau d'interconnexion qui relie physiquement les processeurs entre eux. Le coût de communication entre deux processeurs différents dépend de la position de ces deux processeurs dans le réseau.

La machine à mémoire distribuée a un grand potentiel de puissance de calcul parce qu'il n'y a guère de limitation pratique au nombre de processeurs. Chaque processeur a aussi un accès rapide à sa propre mémoire. De plus, ce type de machine peut être réalisé via la conception de grappe des machines (cluster en anglais) et Le premier point faible de la machine à mémoire distribuée est qu'elle est plus difficile à programmer que la machine à mémoire partagée. Le programmeur devra gérer lui-même les communications. Cela pourrait engendrer des risques d'erreurs. La complexité des algorithmes est source de perte de performances. Les performances seront parfois possibles qu'au prix d'un réseau d'interconnexion coûteux. L'administration du système sur une grappe de machines doit être prise en compte dans le coût global.

II.1.3 Modèles de Programmation parallèle

II.1.3.1 Modèle Data-Parallel

L'un des modèles d'algorithmique parallèle le plus simple est le modèle à parallélisme de données (data-parallel en anglais). L'idée de ce modèle consiste à subdiviser un problème global en tâches se basant sur le partitionnement des données. Chaque tâche, associée à une partie de données, effectue les mêmes opérations (voir Figure 2.6). Le programmeur devrait garantir l'équilibre de charge des données.

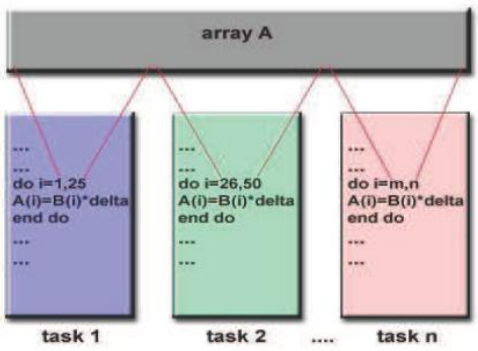


Figure 2.6 – Modèle de programmation de "Data-Parallel"

La programmation data-parallel consiste à rédiger un programme avec des structures de données parallèles de type tableau. Les algorithmes de ce modèle peuvent se baser sur les deux paradigmes : le paradigme de l'espace d'adresse commune et le paradigme de passage de messages. Dans un paradigme à passage de messages, la partition de l'espace d'adresse peut permettre un meilleur contrôle du placement, et peut alors offrir une meilleure manipulation de la localité. D'autre part, l'algorithme, dans le paradigme de l'espace d'adresse commune, peut être facilement programmé,
surtout si la distribution des données est différente dans les différentes phases de l'algorithme (voir [Grama 03]).
Ce modèle s'adapte bien aux problèmes dont le type de données est un tableau d'éléments, tels que les matrices, les images, etc...

II.1.3.2 Modèle concurrent à mémoire partagée

Le modèle de programmation parallèle à mémoire partagé est le modèle le plus simple à comprendre parce qu'il est similaire à la programmation des systèmes d'exploitation. La programmation dans ce modèle se fait à travers des extensions aux langages de programmation existants, les systèmes d'exploitation, et les bibliothèques.
Elle consiste en trois types principaux de routines qui permettent :
1. la création de la tâche ;
2. la communication entre les tâches ;
3. la synchronisation.

Dans le modèle concurrent à mémoire partagée, les tâches se partagent un espace d'adressage commun dans lequel elles peuvent lire et écrire de manière asynchrone (voir Figure 2.7). Quelques mécanismes, tels que le blocage ou des sémaphores, peuvent être utilisés pour le contrôle d'accès à l'espace d'adressage commun.

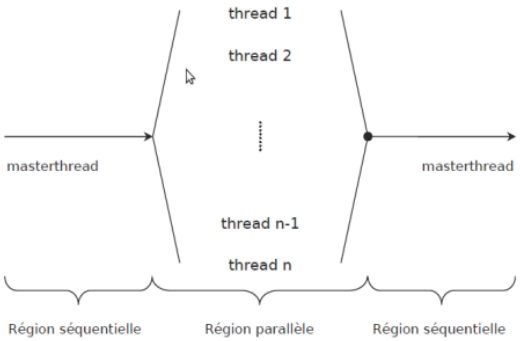


Figure 2.7 – Modèle de programmation à mémoire partagée

II.1.3.3 Modèle concurrent à passage de messages

Le modèle de programmation à passage de messages a été conçu pour les architectures parallèles ne possédant pas de mémoire partagée. L'absence de mémoire partagée engendre deux conséquences importantes pour le programmeur. Dans ce modèle (voir Figure 2.8), la mémoire est fragmentée et le programme n'a pas une vision complète de l'espace mémoire. Cette répartition des données sur les différents processeurs doit être gérée par le programmeur. Chaque processus exécute éventuellement des parties différentes d'un programme. Toutes les variables du programme sont privées et résident dans la mémoire locale de chaque processus.

Les processus communiquent par envois explicites de messages contenant des données calculés par certains et utiles à d'autres. Les messages contiennent en général des données.

À la base, un système d'échange de message consiste en :

– un schéma d'adressage qui repère les processus par rapport aux autres.

– deux primitives de base (par exemple send/receive).

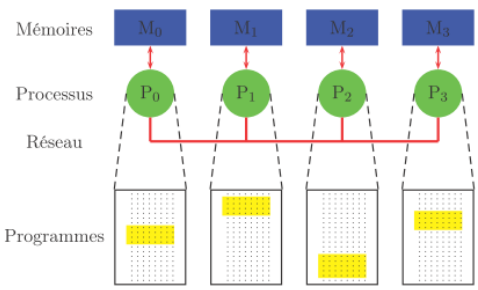


Figure 2.8 – Modèle de programmation à passage de messages

II.1.3.4 Modèle Multi-Threads

Dans le modèle multi-threads, un modèle très proche du modèle à mémoire partagée présenté dans Section II.1.3.2, un processus est défini comme un programme en cours d'exécution d'une séquence d'instructions. Il peut être décomposé en ce qui s'appelle threads, un fil d'instructions à l'intérieur d'un processus. Les threads d'un même processus se partagent l'espace d'adressage du processus. Ils possèdent leur propre compteur ordinal, leur propre pile et sont ordonnancés par le système d'exploitation concurrent. les threads ont aussi un cycle de vie semblable à celui d'un processus.

Ce modèle s'adapte bien à la programmation du côté serveur pour des applications client-serveur. Dans ces applications, un serveur doit simultanément répondre aux requêtes de multiples clients. De plus, le nombre de clients varie dans le temps. C'est-à-dire que les clients se connectent au serveur et se déconnectent dynamiquement. Dans certains cas, les programmes utilisant des threads sont plus rapides sur les machines comportant plusieurs processeurs. En plus, le partage de certaines ressources entre les threads permet une communication plus efficace entre les différents threads d'un processus. La programmation utilisant des threads est toutefois plus difficile, l'accès à certaines ressources partagées doit être restreint par le programme lui-même. De plus, elle repose des questions concernées sur les temps de création de processus et de commutation de contexte. Ensuite, une exigence nécessaire, pour le partage de données communes entre les processus, n'est pas assurée. Cela est très important dans plusieurs applications. Enfin, le programmeur n'a pas de moyen efficace pour gérer l'ordonnancement des processus.

II.1.4 Exemples de librairies facilitant la parallélisation

II.1.4 .1 MPI (Message Passing Interface)

En juin 1994, le forum MPI (Message Passing Interface) a défini un ensemble de sous-programmes de la bibliothèque d'échange de messages MPI. Une quarantaine d'organisations y ont participé. Cette norme, MPI-1 (Le coeur de la bibliothèque), vise à la fois la portabilité et la garantie de bonnes performances. En juillet 1997, la norme MPI-2 a été publiée pour

l'extension et le support C++ et f90. Elle permet la gestion dynamique de processus, la copie mémoire à mémoire, la définition de types dérivés et MPI-IO.

Une application MPI est un ensemble de processus qui exécutent des instructions indépendamment les unes des autres et qui communiquent via l'appel à des procédures de la bibliothèque MPI. MPI nous permet de gérer l'environnement d'exécutions, les communications point à point, les communications collectives, les types de données dérivées, la topologie d'interconnexion des processus (grilles, arbres,...), et les groupes de processus et les groupes de communicateurs.

Parmi l'ensemble des implémentations de MPI, les trois grandes distributions les plus populaires sont :

– MPICH (http://www.mcs.anl.gov/research/projects/mpich2/)

– Open-MPI (http://www.open-mpi.org/)

– LAM-MPI (http://www.lam-mpi.org/)

II.1.4.2 OpenMP (Machines à mémoire partagée)

OpenMP (http://openmp.org/wp/) amène aujourd'hui une interface standard de haut niveau pour une programmation parallèle de type SPMD (Single Program Multiple Data) sur les machines à mémoire partagée ou au moins virtuellement partagée. OpenMP se base sur les techniques du multi-threading, elle est considérée comme l'un des grands standards au service du calcul scientifique.

OpenMP peut être supportée par plusieurs langages (C, C++ et Fortran) et disponible sur plusieurs plate-formes (Linux, Windows, OS X, ...). Les standards d'OpenMP datent de 1997, ceux d'OpenMP2 de 2000. OpenMP est un ensemble de procédures et de directives de compilation identifiées par un mot clef initial $OMP visant à réduire le temps de restitution lors de l'exécution d'un programme sans en changer la sémantique. Elle est aussi utilisée afin de paralléliser un code sur une architecture SMP (Shared Memory Processors en anglais).

II.2 Les cartes graphiques et le paradigme parallèle

II.2.1 Introduction et motivations

Au sein d'une architecture informatique, les processeurs centraux, CPU, ont originellement la charge de l'ensemble des calculs. Lorsque la nécessité de traiter des données

de large volume (images, vidéo ou son) est apparue dans les années 80, leurs architectures se sont adaptées en proposant des jeux de fonctionnalités dédiées à ce traitement multimédia : MMX pour les processeurs de marque Intel, ou bien 3DNow ! pour ceux d'AMD, ont contribué au succès de ces gammes de processeurs. Poussée par l'industrie vidéo-ludique, ayant récemment dépassé en terme de budget l'industrie cinématographique, l'émergence d'un type de matériel informatique dédié aux traitements graphiques déchargeant le CPU, nommé GPU, Graphics Processing Unit, est née, il y a une dizaine d'années, du besoin de plus en plus grand de spécialisation des architectures des processeurs, voire de multiplication de ceux ci. Les GPU sont ainsi des processeurs portés par une carte annexe, prenant à leur charge le traitement des images et des données 3D, ainsi que l'affichage.

L'implantation de ce type de matériel s'est aujourd'hui largement répandue, et a pour conséquence inattendue leur utilisation dans des domaines non graphiques, en particulier en simulation numérique. C'est ce type d'utilisation détournée que l'on nomme GPGPU, General Purpose computing on Graphics Processing Units, ou Programmation Générique sur Carte Graphique.

L'engouement de la communauté pour cette méthode de programmation est tel qu'aujourd'hui, ces cartes graphiques sont devenues de véritables outils computationnels professionnels. Etant architecturalement prévus pour un traitement massivement parallèle des données, ils devraient devenir dans un avenir très proche de véritables coprocesseurs utilisés par tout type d'application. Les raisons d'un tel engouement pour la programmation GPGPU sont très nombreuses, nous nous proposons d'en citer les principales.

Puissance de calcul Les besoins toujours plus importants de réalisme pour le rendu d'images demandent une puissance de calcul croissant continuellement et ont naturellement poussé les industriels à multiplier les capacités matérielles de ces cartes. Que ce soit au niveau du nombre de processeurs parallèles contenus sur les cartes graphiques, du nombre de transistors, de la finesse de gravure ou de la quantité de mémoire disponible, leur évolution a été spectaculaire. La figure 2.9, présente l'évolution comparée des puissances brutes de calcul en terme de GFLOPS, entre les CPU Intel et les GPU NVidia.

Calcul parallèle massif Contenant jusqu'à 240 processeurs, les GPU sont conçus pour exécuter des tâches massivement parallèles, jusqu'à plusieurs milliers de threads. Pour cette raison, les GPU pourraient s'apparenter à des supercalculateurs débarrassés de structure complexe plutôt qu'à des CPU multi-cœurs, capables de traiter seulement quelques threads simultanément. A l'avenir, cela pourrait changer, mais à l'heure actuelle, en privilégiant la rapidité de calcul, les GPU sont préférables.

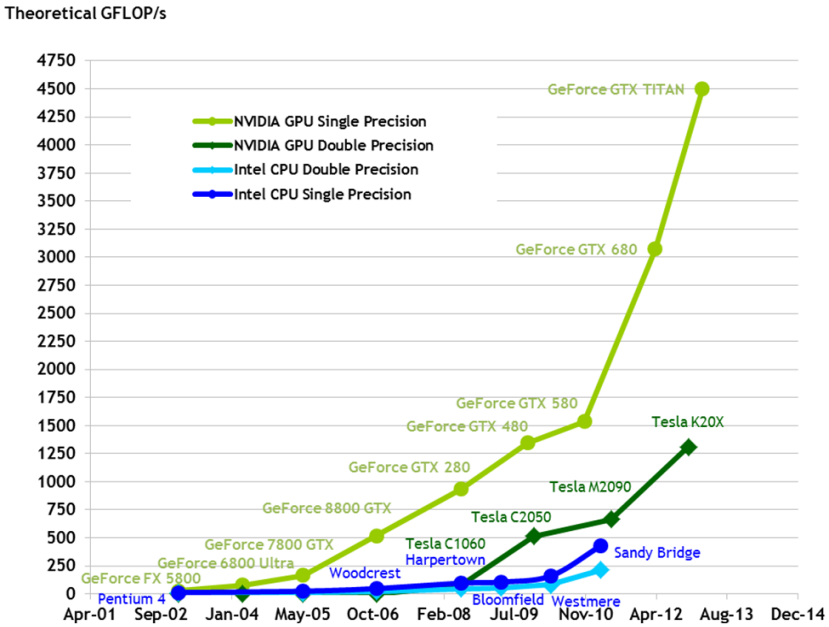


Figure 2.9 – Puissances de calcul brutes comparées entre GPU NVidia et CPU Intel de 2003 à 2014.

Flexibilité Auparavant simplement paramétrables, les cartes graphiques permettent aujourd'hui une programmation flexible et avancée, en proposant de plus en plus de contrôle sur les caractéristiques toujours plus nombreuses, telle l'utilisation de types de données (en particulier le calcul flottant sur 32bits), de structures et d'instructions (branchements conditionnels) de plus en plus complexes. Il existe même dorénavant des outils de débogage destinés à certains langages utilisables en GPGPU. La programmation d'un GPU est ainsi de plus en plus semblable à une programmation séquentielle classique.

Plusieurs GPU GPU, et en particulier les haut-de-gamme, peuvent être combinés en cluster, démultipliant encore la puissance de calcul. C'est le cas de la gamme Tesla, proposées par NVidia, qui associe jusqu'à 4 GPU et 16Go de mémoire dans un unique châssis, portant à 4 TFLOPS (4000 GFLOPS) la puissance de calcul de ce supercalculateur.
Plusieurs calculateurs de ce type peuvent, de plus, être associés par un réseau classique.

Prix et disponibilité Pour environ 400 euro , il est possible de se procurer les GPU les plus puissants existants sur le marché actuel. Ces très faibles prix sont le résultat de l'existence d'un marché énorme et d'une concurrence rude entre les deux plus importants fabricants de cartes graphiques, NVidia et ATI, dont la cible commerciale principale est le grand public, pour la pratique des jeux vidéo. La composition d'un cluster à utilisation professionnelle est donc très raisonnablement envisageable. De plus, par ces faibles coûts, il est aujourd'hui rare

pour une station de travail ou même un ordinateur grand public de ne pas disposer d'un GPU performant.
Ces arguments tendent à prouver que la puissance des GPU, longtemps négligée, est prête à être pleinement exploitée, dans tout domaine computationnel.

II.2.2 Eléments de parallélisme

Les capacités parallèles des GPU permettent d'exécuter un partitionnement d'une tâche complexe en une multitude de tâches élémentaires, pouvant être réalisées par ses différents processeurs travaillant simultanément, rendant l'exécution globale bien plus rapide qu'une exécution séquentielle. Des algorithmes de domaines variés (météorologie, finance, traitement d'image. . .). On se propose de donner ici les définitions de quelques éléments de calcul parallèle, non spécifiques aux GPU.

II.2.2.1 Caractère SIMD/MIMD

Les architectures parallèles, composées en général d'un grand nombre d'unités de calcul, exploitent fortement les modes de fonctionnement SIMD / MIMD, autorisant l'exécution simultanée de plusieurs parties de code. Ces modes sont deux de ceux référencés dans la taxonomie de Flynn, classifiant les différents types d'architectures des systèmes informatiques parallèles toujours d'actualité.
Le modèle SIMD est représenté en figure 2.10. Une exécution d'un code en SIMD peut être synchrone (on attend qu'une partie du code ait été exécuté sur toutes les unités de calcul. On parle de SIMD vectoriel, c'est par exemple le cas pour les GPU) ou asynchrone (chaque unité de calcul progresse indépendamment des autres. C'est alors un SIMD parallèle).
Les instructions SIMD sont accessibles dans de nombreux processeurs professionnels et grand public depuis 1997 : MMX pour Intel, 3DNow !] pour ATI, les jeux SSE (jusqu'à SSE4) pour les processeurs de type x86, AltiVec pour Apple, IBM et Motorola.
Elles sont parfaitement adaptées, par exemple, au traitement de signal, particulièrement le traitement d'image, dans lequel on fait subir à chaque donnée élémentaire, le pixel, le même traitement, de façon indépendante. Dans ces cas, l'exécution du code est en général bien plus rapide qu'une implémentation classique SISD (Single Instruction on Single Data), dans lequel les instructions sont exécutées exclusivement séquentiellement. Cependant, il existe des

contreparties aux processeurs SIMD : leur programmation est souvent malaisée, un algorithme n'est pas forcément exécutable sur une machine SIMD, la gestion des registres est plus complexe,. . .

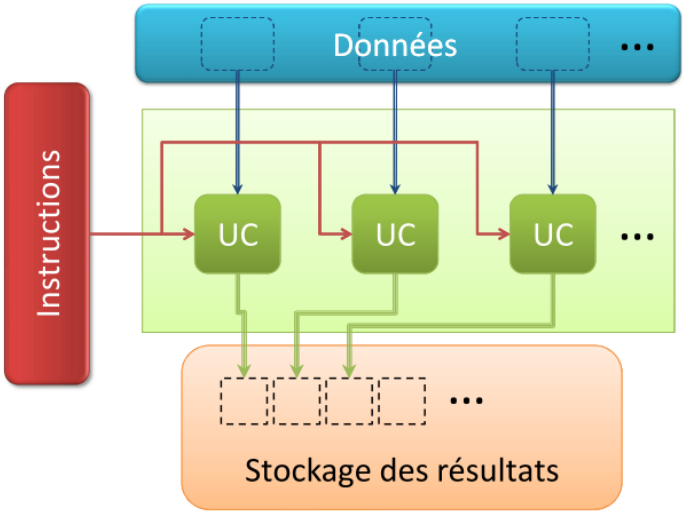


Figure 2.10 – Le modèle parallèle Single Instruction Multiple Data

Pour le modèle MIMD représenté en figure 2.11, chaque unité de calcul reçoit sa propre liste d'instructions et l'exécute sur ses propres données. Les machines MIMD sont plutôt destinées au monde professionnel. Les Connections Machines 5, de Thinking Machines Corporation et les Transputers du défunt Inmos en sont deux des représentants.

On distingue deux types d'accès mémoire dans ce mode :

–*Mémoire distribuée* : chaque unité de calcul possède sa propre partie mémoire et ne peut pas accéder à celles des autres. Une communication est possible entre processeurs par le biais de messages ou d'appels de procédures RPC, mais nécessite de connecter ces unités MIMD de façon adéquate, sur le modèle d'une grille, par exemple.

–*Mémoire partagée* : aucune des unités de calcul n'a de mémoire propre mais toutes peuvent accéder à une même mémoire globale.

Un changement effectué par une unité dans cette mémoire est donc visible depuis les autres unités. Pour éviter les problèmes de synchronisation, les principes classiques d'exclusion sont utilisés : sémaphores, mutex....

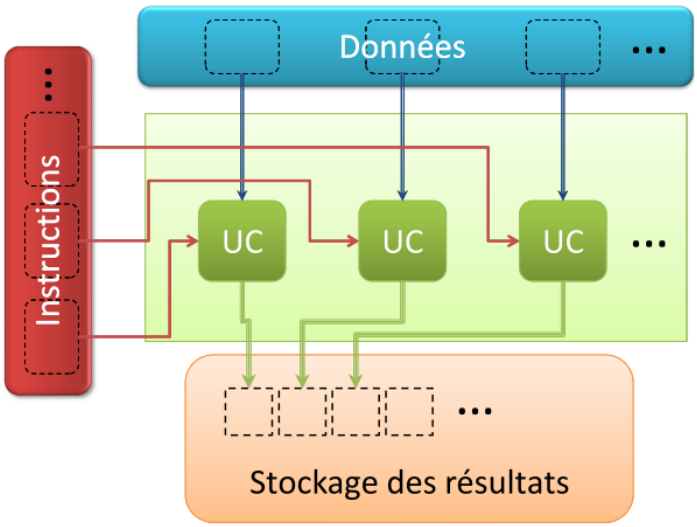


Figure 2.11 – Le modèle parallèle Single Instruction Multiple Data

II.2.2.2 PRAM

On désigne par PRAM, ou Parallel Random Access Machine, un modèle de référence des machines à partage de mémoire, sur lesquels peuvent s'exécuter des algorithmes parallèles (c'est le cas des GPU). Trois modèles sont distingués en fonction de leurs moyens d'accès à la mémoire, liés aux notions de gather et scatter (détaillées dans la section II.2.2.3) :

1. EREW, Exclusive Read Exclusive Write : chaque processeur ne peut lire ou écrire à un endroit mémoire que si aucun autre n'y accède au même moment ;

2. CREW, Concurrent Read Exclusive Write : chaque processeur peut lire à n'importe quel endroit en mémoire, mais plusieurs processeurs ne peuvent écrire simultanément au même endroit.

3. CRCW, Concurrent Read Concurrent Write : chaque processeur peut lire et écrire où il le souhaite en mémoire. Dans ce cas, on distingue trois sous-cas :

- CRCW commun : si plusieurs processeurs écrivent la même valeur au même endroit, l'opération réussit. Sinon, elle est considérée illégale ;
- CRCW arbitraire : si plusieurs processeurs écrivent au même endroit, un des essais réussit, les autres sont avortés ;
- CRCW prioritaire : si deux processeurs écrivent au même endroit, leur rang de priorité détermine le seul qui réussit.

II.2.2.3 Les opérations : Gather et Scatter

Gather et Scatter désignent la possibilité, pour une unité de traitement, de lire, respectivement d'écrire, des données à différents endroits de la mémoire, adressés indirectement. Gather et scatter sont des opérations fondamentales dans tout traitement informatique. La figure 2.12 illustre ces deux principes. Dans un langage ressemblant au C, une opération de lecture type gather s'écrit $u = d[i]$, où d est une structure de données stockée en mémoire, i un index, et u une variable de stockage. Inversement, l'opération de scatter s'écrit $d[i] = u$.

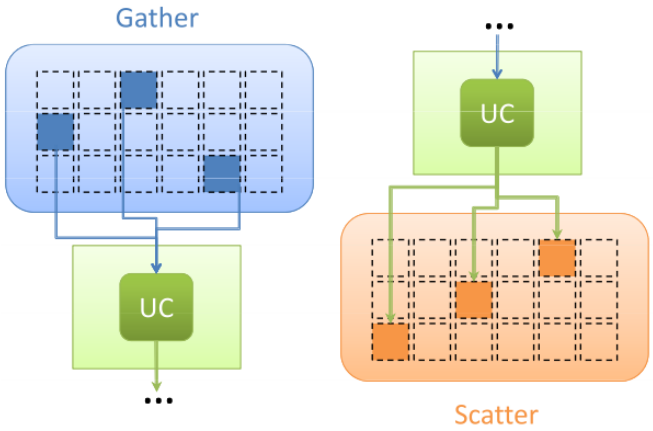


Figure 2.12 – Le "gather" et le "scatter"

II.2.3 Historique des GPU et Organisation interne

Depuis l'apparition dans les années 80 des accélérateurs graphiques matériels, leur évolution n'a cessé de s'accélérer. Les premières cartes graphiques accessibles au grand publiques sont apparues dans le milieu des années 90, avec en particulier les Voodoo Graphics. C'est NVidia qui a introduit le terme GPU, remplaçant le terme VGA Controller, alors insuffisant pour désigner l'ensemble des possibilités offertes par ces cartes.

L'industrie du jeu vidéo, en particulier, n'a depuis cessé de motiver le développement de nouveaux processeurs dédiés au traitement graphiques. Cette évolution peut être catégorisée en différentes générations, selon leurs capacités, sur lesquelles nous nous proposons de dévoiler quelques aspects. Une description plus détaillée de l'architecture des dernières générations, permettant de comprendre leur utilisation dans nos applications, est ensuite proposée.

II.2.3.1 Les différentes générations de GPU

Les forces ayant poussé les industriels à améliorer sans cesse ces cartes graphiques sont dues principalement à un aspect économique de compétitivité, pour faire face à un appétit de plus en plus grand de complexité visuelle ou de réalisme dans les représentations cinématographiques ou les simulations vidéo-ludiques, poussé par une volonté assurément humaine de divertissement.

Suivant la loi de Moore, faite pour les CPU qui ne la respectent plus aujourd'hui, et stipulant que le nombre de transistors sur un processeur double tous les 18 mois, les architectures des GPU ont évolué depuis une décennie, franchissant parfois des sauts technologiques importants.

Nous nous proposons ici de classer ces architectures en différentes générations.

Avant les GPU Dans les années 80, des sociétés comme Silicon Graphics ou Evans and Sutherland proposèrent des solutions matérielles extrêmement couteuses, réservé à quelques professionnels spécialisés, et ne pouvant effectuer que quelques opérations graphiques très simples comme des transformations de vertex ou des applications de textures. De façon générale, le reste des opérations graphiques étaient accomplies par le CPU.

Première génération : La première génération de GPU à proprement parler a débuté avec l'arrivée des Voodoo Graphics de 3dfx Interactive en 1996, et dura jusqu'en 1999. Les principales cartes de cette génération sont les TNT2 de NVidia (architecture NV5), les Rage d'ATI et les Voodoo3 de 3dfx. Les premières opérations graphiques disponibles sont la rasterization de triangles et l'application de texture. Ces cartes implémentent également le jeu d'instruction de DirectX 6, API (Application Programming Interface, Interface de Programmation) alors standard. Cependant, elles souffrent de certaines limitations, principalement la grande faiblesse du jeu d'instructions mathématiques et l'impossibilité de transformer matériellement des vertex, opération lourde encore à la charge du CPU.

Deuxième génération : Les premières GeForce 256 de NVidia (NV10) font leur apparition en 1999, à peu près en même temps que les Radeon 7500 d'ATI (architecture RV200) et Savage 3D de S3. Ces cartes permettent maintenant une prise en charge complète de la transformation des vertex et du calcul des pixels (Transform and Lightning, T&L). Les deux API principales, DirectX 7 et OpenGL, sont maintenant supportées par ces cartes.

Les améliorations apportées au jeu d'instructions rendent ces cartes plus facilement configurables, mais pas encore programmable : les opérations sur les vertex et les pixels ne peuvent être modifiés par le développeur.

Troisième génération : Dès cette génération, les constructeurs NVidia et ATI se partagent la quasi-totalité du marché, NVidia ayant fait l'acquisition de 3dfx. Les Geforce 3 (NV20) en 2001 et GeForce 4 Ti (NV25) en 2002 de NVidia, la Radeon 8500 d'ATI (R200) en 2001 forment cette génération de GPU, permettant enfin au développeur de diriger la transformation des vertex par une suite d'instructions qu'il spécifie. Néanmoins, la programmabilité des opérations sur les pixels n'est toujours pas possible. DirectX 8 et quelques extensions à OpenGL permettent tout de même une plus grande souplesse de configuration dans le traitement de ces pixels.

Quatrième génération : Courant 2002, ATI propose sa Radeon 9700 (R300), et NVidia sa GeForce FX (NV30) à la fin de la même année. Ces deux cartes, en plus de supporter le jeu d'instructions DirectX 9 et de nouvelles extensions OpenGL, apportent de plus la possibilité de programmer le traitement des pixels. C'est à partir de cette quatrième génération de cartes que les premières opérations GPGPU ont pu se faire.

Cinquième génération : Les GeForce 6 (NV40, avril 2004) et GeForce 7 (G70, juin 2005) de NVidia, les Radeon X800 (R420, juin 2004) et X1800 (R520, octobre 205) et dérivées composent cette génération. Ces cartes permettent quelques fonctions intéressantes, en particulier en calcul GPGPU : l'accès aux textures lors de la transformation des vertex, le rendu dans différentes textures (Multiple Render Target, MRT) et le branchement dynamique, uniquement pour la transformation des sommets, améliorant grandement la vitesse d'exécution. Le traitement des pixels ne supporte pas ce branchement dynamique.

Sixième génération : C'est la génération en plein essor actuellement, équipant la plupart des ordinateurs sur le marché. Ses limites sont plus floues, chaque constructeur apportant ses innovations propres. Elle est composée des GeForce 8 (G80) et GeForce 9 (G92), lancées respectivement en 2006 et 2008, apportant des modifications dans la façon de concevoir le pipeline graphique. Entre la transformation des vertex et la rasterization des primitives (leur transformation en fragments, ou pixels), une étape supplémentaire est insérée, permettant de modifier la géométrie du maillage des primitives : insertions et suppression de vertex sont possibles, ce qui ouvre de nouvelles voies au calcul GPGPU. L'ajout du type int utilisable en interne (registres) et surtout en externe (textures) a également contribué à élargir le nombre d'applications possibles. De plus, sur les cartes GeForce 8, il n'existe plus de différence physique entre les différents processeurs. L'architecture matérielle modifiée est dite unifiée, cette caractéristique étant exploitée par le langage de programmation CUDA, de NVidia, utilisable avec des cartes à architectures G80 ou supérieures. La série GeForce 9 ne connait

pas un succès véritable, ayant moins de mémoire et ses performances étant égales voire inférieures aux cartes équivalentes de la série GeForce 8, pour un prix semblable. Les séries de cartes d'ATI de cette génération sont les Radeon HD2000 et Radeon HD3000 (R600) , lancées en 2007, pour contrer la série GeForce 8. Malgré quelques améliorations intéressantes proposées par ce constructeur (support de DirectX 10.1, de la norme Shader 4.1 par exemple), ces cartes peinent à s'imposer à cause d'un prix trop élevé, de pièces bruyantes et de puces chauffant beaucoup.

Septième génération : Les GeForce 200 (G200), lancées en juin 2008, apportent des améliorations principalement techniques : augmentation de la mémoire disponible, du nombre de processeurs, des fréquences mémoire et GPU, de la bande passante par élargissement du bus de données, pour des performances annoncées jusqu'à deux fois supérieures aux séries précédentes. Pour ATI, les Radeon HD4000, en juin 2008 également, sont censées rivaliser avec la série GeForce 9, mais les performances des modèles haut de gamme les mettent au même niveau que les GeForce 200, proposant les mêmes types d'améliorations techniques, mais affichant des tarifs plus attractifs et une consommation électrique revue à la baisse.

Rétrocompatibilité. Il est important de noter que les programmes écrits pour un type ou une génération de cartes restent utilisable avec les générations futures de cartes, même si toutes les capacités ne sont plus nécessairement exploitées. C'est, par ailleurs, un problème important de développement que de devoir faire avec les différentes générations de cartes présentes sur le marché.

Huitième génération : La GeForce® GTX™ TITAN Z est un véritable monstre de puissance. Cette nouvelle carte graphique GeForce, la plus rapide au monde, accélère les performances des PC conçus pour être les plus puissants au monde. Avec son architecture bi-GPU couplée à 5760 cœurs CUDA et 12 Go de mémoire dédiée.

II.2.3.2 Organisation d'un GPU

Deux aspects importants permettent de caractériser l'organisation des GPU : le contenu matériel orienté calcul parallèle et son organisation adéquate en pipeline.

II.2.3.2.1 Composants matériels

La carte graphique est un des organes de l'ordinateur, chargé du traitement graphique et de l'affichage. Elle se compose d'une zone mémoire et de processeurs, auxquels viennent s'ajouter divers registres et chipsets de communication.

Ce qu'on appelle GPU, au sens strict, est l'ensemble des processeurs graphiques contenus sur cette carte, c'est-à-dire les unités opérant les calculs. Par abus de langage, on nommera également GPU la carte entière.

La mémoire disponible varie aujourd'hui jusqu'à 768Mo, cadencée à 1080MHz, accessible en lecture et en écriture par les processeurs de la carte mais aussi par le CPU.

Les GPU embarquent jusqu'à 128 (G80) ou 240 processeurs de flux chacun (G200), cadencés jusqu'à 1500MHz, et répartis en différentes catégories, abordés dans la section II.2.3.2.2. Ces processeurs sont nommés processeurs de flux en raison de leurs natures : ils sont en effet capables de traiter, de façon parallèle, un ensemble de données élémentaires présenté sous la forme d'un flux. Chacun de ces processeurs fonctionne en mode SIMD parallèle /CREW, il reçoit donc un ensemble d'instructions qui sera exécuté sur la totalité des données du flux. Ils sont de plus tous capables de gather, mais pas de scatter : ils peuvent accéder à n'importe quelle adresse mémoire en lecture, mais pas en écriture (voir sections II.2.2.1 et II.2.2.3).

Les processeurs d'un GPU sont organisés en pipeline que nous nous proposons de décrire.

II.2.3.2.2 Pipeline graphique actuel

Quelques définitions sont utiles pour pouvoir expliquer cette section.

Un ***vertex*** est un point géométrique 3D, sommet d'un ou plusieurs polygones.

Un ***fragment*** est une partie élémentaire d'une scène en cours de rendu qui pourrait occuper l'espace d'un pixel dans l'image finale. La différence entre pixel et fragment est une vue d'esprit : alors que le pixel est un "point" de couleur visible par l'utilisateur, le fragment est le "point" de couleur que le programme manipule, provenant de différentes textures, et dont le traitement produira un pixel.

Un ***shader*** est un programme à vocation graphique, généralement court, que le développeur peut spécifier en différents langages, maniant des vertex ou des fragments et permettant de contrôler un sous-ensemble des processeurs d'un GPU.

Un pipeline est une séquence ordonnée de différents étages. Chaque étage récupère ses données de l'étage précédent, effectue une opération propre et renvoie ses résultats à l'étage suivant. Un pipeline est dit rempli lorsque tous ses étages sont mis à contribution simultanément, c'est son utilisation optimale, diminuant le nombre d'étapes globales d'exécution d'un algorithme par rapport à une version séquentielle classique. Le pipeline des

GPU actuels est représenté figure 2.13. Il est composé de trois étages programmables, pilotés par les shaders, et d'un étage non programmable, le rasterizer.

Pour traiter la géométrie d'une image, les données sont premièrement envoyées au GPU. Ces données sont les primitives géométriques de la scène à rendre, décrite via les commandes d'une API graphique, et représentés sous la forme d'un flux de vertex. Ces vertex sont les données d'entrée du premier étage du pipeline, le Vertex Shader. Celui ci a pour vocation de modifier les différents attributs des vertex : il peut les changer et leur en donner de nouveaux, comme couleur et normale. Le flux sortant est de même taille et est composé des mêmes vertex modifiés. Il a accès à la mémoire de la carte en lecture et peut y écrire ses résultats. Sa principale utilité est de déplacer les objets pour donner l'impression de mouvement d'une frame à une autre.

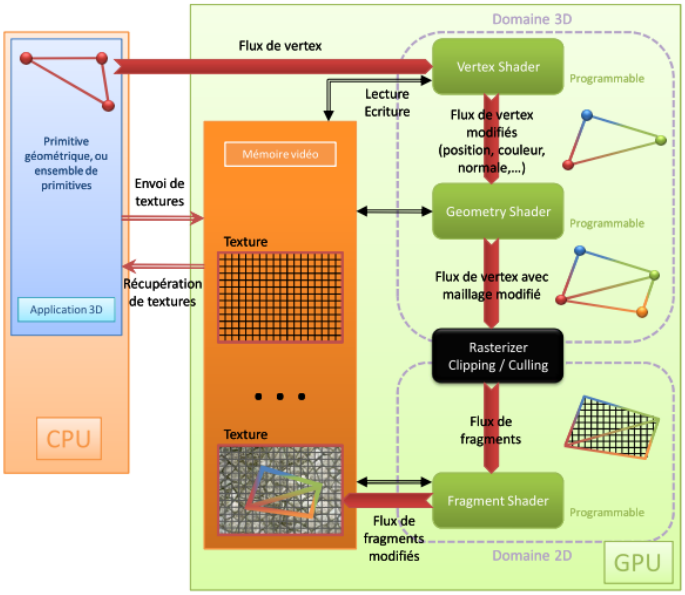


Figure 2.13 – Pipeline GPU

Le flux sortant du précédent shader est le flux d'entrée pour l'étage suivant du pipeline, le Geometry Shader. Ce shader permet de modifier la géométrie de la scène, ici le maillage géométrique passé à la carte. Il peut en effet ajouter ou supprimer des vertex et en modifier les attributs.

Le flux de sortie contient un ensemble de vertex de taille potentiellement différente à celle du flux d'entrée. Il a également accès à la mémoire en lecture, et en écrire pour ses résultats. Ce shader est en particulier utilisé pour les méthodes de raffinement de maillage ou d'augmentation de détails sur les modèles géométriques, comme le Displacement Mapping, technique permettant de créer le relief en surface d'un objet en déplaçant certains points de cette surface en fonction de valeurs contenues dans une texture de hauteur (displacement

map). L'étage suivant dans le pipeline est le Rasterizer. A partir du maillage composé des vertex sortant du shader précédent, la géométrie finale est projetée sur une grille de la taille de l'image de sortie, et les primitives géométriques sont divisées en fragments. C'est à cet étage du pipeline qu'on réalise également quelques opérations supplémentaires améliorant
la vitesse d'exécution des calculs suivants, telles que clipping (suppression des objets extérieurs au cône de vision donc non visibles) ou back-face culling (suppression de polygones suivant leur orientation par rapport à la caméra : un polygone présentant son "dos" à la caméra n'est pas censé être visible). Les fragments résultants possèdent des coordonnées, correspondant à la position finale dans l'image. Le flux de fragment sortant est ensuite dirigé vers le dernier étage du pipeline. Le Fragment Shader est le troisième et dernier étage programmable du pipeline. C'est le shader le plus important car pour chaque fragment de son flux d'entrée, il lui attribue sa couleur finale, en fonction de l'éclairage, des réflexions et réfractions lumineuses et de diverses techniques de rendu pouvant interroger des textures tierces. Comme les shaders précédents, il peut consulter la mémoire de la carte mais ne peut y écrire n'importe où. Il a seulement l'autorisation d'écrire aux coordonnées du fragment qu'il est en train de traiter, coordonnées qu'il ne peut donc modifier.
Le flux de fragments calculés, ou shadés, est écrit en mémoire. Classiquement, la structure accueillant ces données est le framebuffer, directement affiché à l'écran, image ne nécessitant pas un transit supplémentaire par le CPU. Néanmoins, il est possible d'écrire cette image dans une texture, pouvant être utilisée par un autre shader ou récupérée sur CPU.
Jusqu'à la cinquième génération de cartes graphiques, les processeurs embarqués étaient catégorisés, certains servant au traitement de vertex, les Vertex Units, d'autres à celui des fragments, les Fragment Units. Cela a pour inconvénient de pouvoir créer un bottleneck : lorsque les Vertex Units sont surchargés, l'utilisation des Fragment Unit n'est pas optimale et peut même être extrêmement diminuée, et réciproquement. Avec la mise à disposition de la sixième génération de cartes graphiques (GeForce 8), les processeurs ne sont plus spécifiques, ils peuvent maintenant servir aussi bien à la gestion des vertex qu'à celle des fragments. Cela permet de remplir au mieux le pipeline et d'éviter le bottleneck précédent. L'architecture de ces cartes est dite unifiée.

II.2.4 Principes de base pour la programmation en GPGPU

Adapter un algorithme générique pour un calcul sur matériel spécialisé, tel un GPU, demande de prendre en compte quelques aspects spécifiques de ces architectures.

Originairement, les GPU sont destinés au traitement et à l'affichage de graphismes; ils manient de façon naturelle vertex et pixels, sont capables de rasterizer des primitives et d'utiliser quelques structures simples de données tout au mieux. Les libertés dont jouissent les développeurs sur CPU, concernant types complexes de données, gestion de mémoire ou richesse des jeux instructions, sont alors drastiquement réduites par les limitations matérielles inhérentes aux architectures de ces cartes. Il a donc été nécessaire de repenser le portage d'algorithmes génériques sur GPU. Un modèle de programmation classique en GPGPU en a émergé, ainsi que différentes astuces largement employées. Dans cette section, après un tour d'horizon des langages de programmation disponibles sur GPU, nous exposons ce modèle de programmation GPGPU, ainsi que les contraintes et astuces liées à ce type de programmation. Nous continuons avec une rapide présentation de deux langages utilisés en GPU, Cg et CUDA, puis les comparons sur l'implémentation de quelques techniques usuelles de calcul parallèle.

II.2.4.1 Langages

Le but initialement graphique des GPU a bien entendu dirigé les constructeurs à initialement proposer des langages destinés à ce type de calcul, c'est-à-dire dont les instructions utilisent les branchements matériels spécifiques, destinés à des opérations de type *mult-add*, largement
utilisées en synthèse d'images. Même si d'autres langages plus généralistes ont depuis été présentés, le choix reste bien plus restreint sur GPU que sur CPU. On s'intéresse ici à faire un tour d'horizon non-exhaustif des langages existants sur GPU. En plus des langages bas niveau type Assembleur dont il ne sera pas fait mention dans cet exposé, manipulant directement le contenu de registres des unités programmables sur la carte, les langages de plus haut niveaux disponibles sur GPU se distinguent en deux grandes catégories: les Shading Languages, dont les instructions sont principalement destinées aux calculs graphiques, et ceux dont l'objectif
est le calcul générique, que l'on nommera langages GPGPU.

II.2.4.1.1 Shading Languages

Ces langages ont pour point commun de proposer au développeur des jeux d'instructions spécifiques à la génération d'images, ils sont orientés matériel. Ils permettent d'écrire des

shaders. Ceux ci seront exécutés par les unités matérielles spécifiques, Vertex Unit et Fragment Unit, ces deux types d'unités étant unifiés dans les dernières générations de cartes.
Cg, HLSL, GLSL Ces trois langages, respectivement proposés par NVidia, Microsoft et 3DLabs, ont été développés conjointement et sont donc très similaires. Ce sont les shading languages les plus usités. Les syntaxes et sémantiques proposées ont été délibérément fixées proches des instructions du C++, par soucis de simplicité.

II.2.4.1.2 Langages GPGPU

L'utilisation des shading languages pour de la programmation générique n'est pas des plus aisées. Le développeur les employant est contraint d'adapter ses algorithmes à un modèle de calcul basé sur des primitives géométriques et des textures, ce qui n'est pas toujours facile ni même possible. Des moyens pour programmer les GPU sans pour autant avoir besoin de connaissances en graphisme ont naturellement suscité un intérêt grandissant. Différents langages ont émergé de projets industriels ou universitaires.
Tous ces nouveaux langages ont comme dénominateur commun d'être de plus haut niveau que les shading languages, en proposant plus de possibilités et en se séparant totalement des notions graphiques : texture, shader, vertex ou fragment n'y ont plus de sens. Les plus importants langages GPGPU sont présentés dans la suite.
CUDA CUDA [Buck 07, Labatut 06], Compute Unified Device Architecture, est le dernier né des langages de NVidia, pouvant être utilisé à partir de la sixième génération de cartes graphiques de ce constructeur (architectures G80 et plus récentes). Quelques unes de ses particularités sont qu'il soit encastré dans du code C++ en en définissant seulement quelques extensions, de mettre à disposition une mémoire partagée et rapide, ou de supporter différents types d'opérations scalaires, en particulier les opérations sur entiers et bit à bit. C'est aussi le premier langage à exploiter l'unification des shaders sur les architectures G80 de NVidia. Dans le domaine du traitement d'images, il est de plus en plus employé ; Young et Jargstorff proposent dans [Neil 08] quelques implémentations astucieuses d'algorithmes classiques de ce domaine. Une présentation plus détaillée de CUDA est proposée dans la section 1.4.6 (où ça ?).
OpenCL Le très récent OpenCL [Aaftab 08, Neil 08] (Open Computing Language) a été annoncé par Apple au sein du Compute Working Group, formé par le Khronos Group (regroupant 3DLabs, Apple, AMD, NVidia, ARM, Ericsson et d'autres universitaires et industriels). Ces partenaires souhaitent mettre à disposition un langage open source, dans la

veine d'OpenGL et OpenAL [Alexandre 06] et ont pour ambition de faire d'OpenCL le standard libre pour le calcul GPGPU, avec les importants avantages d'être multiplateforme (cartes AMD/ATI et NVidia) et de permettre une programmation homogène BrookGPU, Brook+ [BrookGPU] est une implémentation GPU du langage Brook, tous deux développés par le Stanford University Graphics Lab.

C'est un langage basé sur la gestion de flux de données. AMD/ATI a également proposé Brook+, une amélioration de BrookGPU pouvant être utilisé uniquement sur leurs cartes. Folding@home, projet mondial de calcul distribué simulant les repliements de protéines dans le but d'en tirer des solutions médicales, utilise en partie Brook+. Scout Proposé par le Los Alamos National Laboratory, Scout[Patrick 04] est un langage GPGPU destiné à l'analyse et à la visualisation scientifique, dont beaucoup de techniques sont basées sur l'utilisation de mappings, exprimés sous forme de fonctions mathématiques et transformant des données en images affichables. Scout a notablement été utilisé pour la simulation et la détermination de caractéristiques du courant côtier El Ninõ.

Accelerator Microsoft Research a présenté Accelerator [David 06], destiné à simplifier la programmation GPGPU en fournissant un modèle de calcul parallèle accessible simplement à travers d'autres langages. Les opérations parallèles y sont compilées à la volée et optimisées pour les fragments shaders.

CGis Le langage CGiS [Nicolas 04], développé par l'Universität des Saarlandes, est un autre langage parallèle, similaire à Brook et Accelerator, manipulant également des objets de type flux de données, mais se démarquant par l'absence de notion de kernel computationnel (équivalent de shader pour du calcul uniquement GPGPU), remplacé par un mécanisme de boucle globale *forall*, chère au calcul parallèle.

II.2.4.2 Modèle de programmation

La programmation GPGPU est basée sur quatre concepts principaux, que nous proposons d'expliciter. On évoque également deux autres façons, liées, d'interpréter la notion de complexité en calcul sur GPU.

II.2.4.2.1 Tableaux = Textures

Sur CPU, une des structures de données la plus utilisée est le tableau $1D$. Les tableaux de dimensions supérieures sont représentés dans un tableau $1D$, les données y étant stockées à la

suite pouvant être accédées par des offsets sur leurs indices. En GPU, les données exploitables sont nécessairement stockées dans des textures (donc en 2D), on y accède par leurs coordonnées. Pour y représenter des structures d'autres dimensions, il est utile de passer par une réorganisation vers un tableau 2D, transmis à la carte dans une texture. Le GPU peut alors lire l'élément (i, j) du tableau en consultant le pixel (i, j) de cette texture. Dans le cas d'un tableau $1D$ stocké dans une texture de taille $N \times N'$, l'élément k a pour coordonnées dans la texture $(k \ mod \ N, E\left(\frac{k}{n}\right))$où $E(x)$ est la fonction partie entière de x. Les tailles maximales pour les textures sont de 8192 × 8192 avec l'API DirectX, ou bien de 4096 × 4096 pour les versions moins récentes. Bien qu'il n'y ait à priori pas de contrainte sur le nombre de textures instanciées simultanément, c'est souvent la quantité de mémoire sur la carte qui va limiter ce nombre, s'élevant aujourd'hui jusqu'à 768Mo par carte grand public, ou 1Go pour des cartes professionnelles.

Chaque élément de texture peut contenir jusqu'à 4 scalaires (correspondant aux canaux rouge, vert, bleu et alpha d'un pixel à afficher). Les opérations scalaires se font simultanément sur ces 4 valeurs. Les types de scalaires utilisables pour les éléments de texture peuvent être entiers ou flottants. Ces éléments peuvent être vectoriels de dimension 1 à 4, voire matriciels de dimensions allant jusqu'à 4 × 4.

II.2.4.2.2 Kernel = Fragment Shader

La programmation parallèle a introduit la notion de kernel computationnel, brique calculatoire de base, équivalent d'une fonction mathématique, pouvant être appliqué à un ensemble de données de façon parallèle. En programmation GPGPU, ces kernels sont les fragments shaders. Un tel shader comporte donc une suite courte d'instructions opérant sur une donnée (ou un petit ensemble de données). Il est à noter qu'aucun vertex shader ou geometry shader n'est en général implémenté. Il est possible de ne pas spécifier de shader, les unités de traitement correspondantes se comportant alors comme des passthrough, ne modifiant aucun attribut des
données.

II.2.4.2.3 Calcul = Rendu graphique

Une fois le fragment shader implémenté, une fois l'environnement convenablement préparé (textures créées aux bonnes dimensions, données d'entrée envoyées à la mémoire

GPU, paramètres supplémentaires assignés, environnement OpenGL ou DirectX correctement initialisé,. . .), l'exécution du calcul décrit dans le fragment shader se fait en utilisant la totalité du pipeline graphique, par invocation du rendu d'un simple rectangle de taille adaptée aux données de sortie. En effet, un tel rectangle rasterizé est comparable à une grille de taille fixée, structure la plus pratique à disposition pour la réalisation de calculs parallèles. Cela justifie la non-modification des sommets, et donc la non-utilisation des vertex et geometry shaders. Ces étapes sont résumées sur la figure 2.14. Harris présente également quelques principes de ce type de programmation dans [Mark 05]. Le lancement d'un rendu avec un shader est également appelé passe du shader.

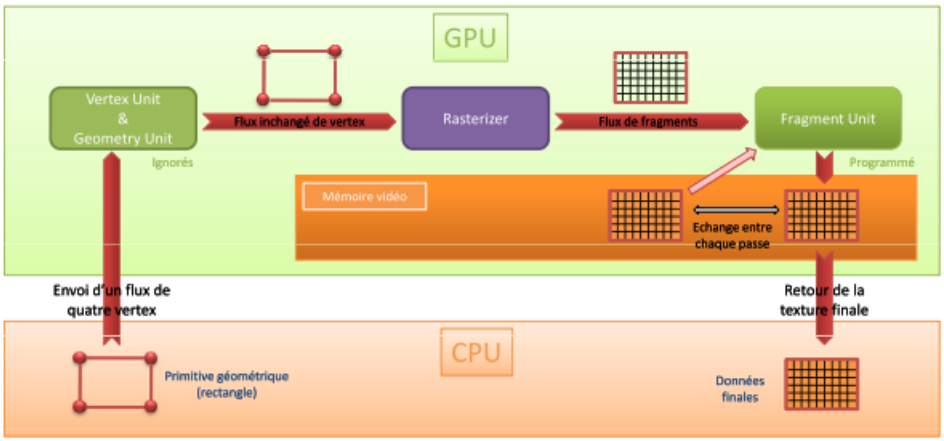


Figure 2.14 – Modèle classique de programmation GPGPU

II.2.4.2.4 Feedback

Lors d'un calcul destiné à faire un affichage à l'écran, l'image générée est stockée dans le framebuffer, buffer dédié à cet effet. Pour un calcul GPGPU, le résultat du calcul n'est pas stocké dans ce framebuffer, mais dans une texture, par un moyen technique nommé Render-To-Texture, RTT [Wynn 04]. Cette texture peut alors être utilisée comme donnée d'entrée pour une passe ultérieure, permettant ainsi de segmenter tout algorithme à implémenter en GPU en différentes passes. Lorsqu'un algorithme nécessite plusieurs passes successives, une astuce simple et classique, appelée ping-pong, est d'utiliser une paire de textures, une pour l'entrée de l'algorithme, une pour sa sortie, que l'on permute entre chaque rendu. Cela permet de ne pas avoir à transférer les données du GPU vers le CPU et réciproquement entre chaque passe.

II.2.4.2.5 Complexité GPGPU

La complexité d'un algorithme parallèle GPU se mesure différemment d'un algorithme séquentiel classique. On peut vouloir se reposer sur le nombre total de fragments à calculer, on parle alors de complexité en fragment C_f . On peut également vouloir déterminer le nombre de rendus nécessaires pour obtenir le résultat final, c'est alors la complexité en rendus C_r. Ces deux complexités sont liées par un facteur multiplicatif, la première étant égale à la seconde multipliée par la taille N des données : $C_f = C_r \times N$.

II.2.4.3 Introduction à CUDA

CUDA, pour Computer Unified Device Architecture, est à la fois un framework et un langage de programmation de NVidia, proche du C++, permettant d'exploiter les capacités des GPU d'architecture G80 et plus de NVidia (à partir des GeForce 8800 et déclinaisons), et exploite les ressources matérielles des GPU d'une façon différente à Cg, en particulier en ce qui concerne la gestion de la mémoire et l'organisation des traitements. C'est aussi le premier langage à exploiter l'unification des shaders : les processeurs de la carte ne sont pas différentiés en unités pour le traitement des vertex ou des fragments, chacun de ceux ci peut être assigné à n'importe quelle tâche.
Avec l'arrivée de CUDA, NVidia a dévoilé, à propos de ses cartes récentes, certaines caractéristiques qu'il n'était pas nécessaire de connaitre auparavant. En CUDA, certaines prennent un sens, c'est le cas de l'organisation des processeurs sur la carte, ainsi que celle de la mémoire disponible par processeur.

II.2.4.3.1 Structures de données et variables disponibles

Etant proche du langage C++, CUDA permet au développeur d'utiliser et de passer en paramètre aux programmes écrits un large panel de structures, en particulier les pointeurs ou les tableaux à plusieurs dimensions, ce qui était impossible en Cg. L'allocation de structures est par exemple possible dans différentes zones de la mémoire, partagée à différents degrés (voir section II.2.4.3.5).

II.2.4.3.2 Organisation matérielle

Dans la section 2.3.2.2, il a été dit que les processeurs d'un GPU se répartissent en deux catégories : vertex unit et fragment unit. Ceci était un abus de langage concernant l'architecture G80 de NVidia, dite unifiée : tous les processeurs sont identiques et peuvent exécuter toutes les instructions destinées à ces traitements. Le langage CUDA exploite cette caractéristique, se détachant totalement des notions de vertex, maillage et fragment. CUDA tire également parti de la disposition des processeurs des architectures unifiées (G80, G92 et G200 pour l'instant). Sur les architectures G80 et G92, ces processeurs sont répartis en 16 groupes de 8 processeurs, ces groupes étant nommés multiprocesseurs. Sur les architectures G200, il existe 10 grappes de 3 multiprocesseurs de 8 processeurs. Ce type de regroupement permet de faciliter les échanges pour un travail à haute fréquence (jusqu'à 1500Hz) des processeurs d'un multiprocesseur. Il est à noter que cette organisation matérielle n'est pas en opposition avec le modèle de pipeline graphique présenté dans la section 2.3.2.2, c'est ce pipeline graphique qui repose sur cette organisation. En effet, dans un schéma de programmation graphique, ou GPGPU avec shading language, chaque processeur est dédié soit au traitement des vertex, soit à celui de la géométrie, soit à celui des fragments, rôle qui peut varier d'un calcul à l'autre. Dans ce cas, ils sont contrôlés par les shaders correspondants. Cela justifie l'abus de langage réalisé.

II.2.4.3.3 Kernels

Contrairement à Cg, CUDA ne dépend d'aucune API graphique tierce. Il propose sa propre API haut niveau, consistant en quelques extensions au langage C et dont la prise en main ne nécessite pas de connaissances approfondies dans le domaine graphique. En effet, CUDA est exclusivement dédié au calcul GPGPU : texture, fragment et pixel n'y ont pas de réelle signification. Les données ne sont plus stockées dans des textures, mais dans des flux, classiquement des tableaux. Les routines pilotant les processeurs sont des kernels comme définis dans la section 2.4.2.2, appliqués à tous les éléments du flux entrant ; ce sont les équivalents des fragment shaders. L'exécution du code se fait par une invocation de kernel, équivalent d'un rendu en Cg, et écrivant ses résultats dans un flux de sortie.

II.2.4.3.4 Organisation des threads

L'exécution de programmes CUDA repose sur la notion de thread, similaire à celle de threads sur CPU, processus légers pouvant s'exécuter en parallèle. Ces threads sont regroupés en warp, des ensembles de 32 threads ; c'est la taille minimale que le GPU peut traiter en SIMD. En pratique, le développeur ne manipule pas ces warps, mais des blocs, qui sont également des réunions de 64 à 512 threads (de 2 à 16 warps), organisés dans un espace de une à trois dimensions. Ces blocs sont enfin regroupés par grid, ou grille, dans un espace de même type. Tous les threads contenus dans une grille sont pilotés par un même kernel. Une invocation de kernel est donc une exécution d'un kernel sur tous les threads d'une grille. Le schéma 2.15 résume l'organisation de ces groupements.
La communication entre les threads d'un même bloc est permise par une mémoire partagée par le bloc. Il est possible de synchroniser tous les threads d'un même bloc.

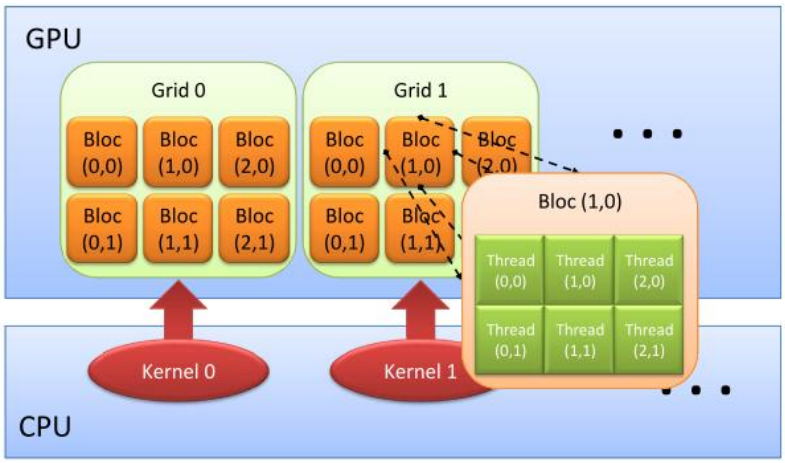


Figure 2.15 – Organisation des unités logiques de traitement pour CUDA

II.2.4.3.5 Organisation mémorielle

Chaque processeur, exécutant un thread, a un accès physique à plusieurs endroits mémoriels. Les deux plus larges zones mémoire sont les mémoires locale sur GPU et globale sur CPU, les processeurs du GPU ayant en effet accès à une partie de la mémoire du reste du système. Ces
accès sont lents (entre 200 et 300 cycles d'horloge) et ne sont de préférence utilisés que lorsqu'ils peuvent être masqués par d'autres calculs. Une partie de la mémoire globale du

GPU peut être mise en cache sur un multiprocesseur, et accessible aux huit processeurs de ce multiprocesseur ; ce sont les caches pour les constantes et pour les unités de texture, de 8Ko chacun, accessibles uniquement en lecture. Chaque multiprocesseur propose également une mémoire partagée entre ses processeurs, de 16Ko, permettant l'échange d'informations rapidement entre eux et économisant la bande passante. Cette mémoire n'est cependant partagée qu'entre les threads d'un même bloc. Enfin, chaque processeur dispose de quelques registres qui lui sont propres. La figure 2.16 résume cette organisation des espaces mémoire utilisée en CUDA. Ce schéma mémoriel est l'un des atouts majeurs de CUDA, car les différents accès en lecture et écriture à ces zones autorisent les opérations de scatter, permettant des portages beaucoup plus simples d'algorithmes sur GPU.

II.2.4.3.6 Exécution

Le nombre de blocs pouvant être exécuté simultanément est conditionné par l'architecture présente, un modèle de GPU disposant de plus ou moins de multiprocesseurs. L'inclusion d'un grand nombre de blocs dans des grilles permet de s'abstraire de cette contrainte matérielle et d'exécuter un kernel sur un grand nombre de threads en une seule invocation.

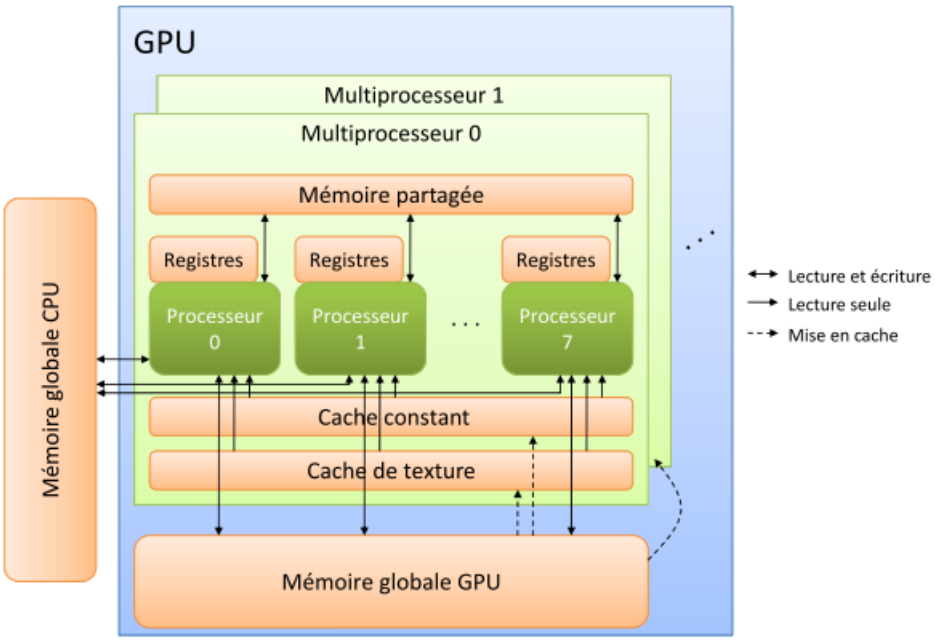


Figure 2.16 – Organisation des mémoires

De plus, CUDA se charge de répartir les ressources disponibles : sur les GPU disposant d'un nombre important d'unités de traitement, les blocs de threads seront exécutés simultanément sur ces unités ; dans le cas contraire, une exécution séquentielle aura lieu.

Chaque grille est assignée à un multiprocesseur. Les 16Ko de mémoire du multiprocesseur sont donc partagés par tous les blocs que ce multiprocesseur exécute. Ce qui reste à la charge du développeur est l'arrangement des threads par bloc et des blocs par grille, en en spécifiant les nombres, généralement dépendant de la taille des données en entrée plutôt que du nombre de processeurs disponibles sur la carte. Le modèle de programmation en CUDA est un une extension du modèle GPGPU présenté en section II.2.4.2 : ce qu'il est possible de faire en programmation GPGPU classique (avec un shading language) est a priori possible en CUDA.

II.2.4.3.7 Exemple de code

Programmer en CUDA, c'est écrire un ou plusieurs kernels et les invoquer à partir d'un programme principal. L'exemple canonique de démonstration du fonctionnement de ce langage est l'addition de deux matrices, que nous prendrons ici carrées. Nous présentons donc ici l'implémentation d'un kernel et du code C++ associé.

Le kernel se définit comme une fonction C++ standard, précédé de la déclaration __global__, indiquant que la fonction est à exécuter sur le GPU et non le CPU. Dans notre cas, on peut l'écrire de cette façon :

```
__global__ void add_matrix(float A[N][N],
                           float B[N][N],
                           float C[N][N])
{
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  int j = blockIdx.y * blockDim.y + threadIdx.y;
  if (i < N && j < N)
    C[i][j] = A[i][j] + B[i][j];
}
```

Le kernel add_matrix prend en argument trois matrices carrées de taille N, A et B étant les deux matrices à additionner dans C. Chaque thread lancé est doté d'un identificateur, threadIdx, a accès à sa position dans le bloc courant, blockIdx, et aux dimensions de ce bloc, blockDim. Les variables threadIdx, blockIdx et blockDim sont des vecteurs de dimension 3, car les threads et les blocs peuvent être organisés sur une, deux ou trois dimensions. Cela permet le lancement d'un calcul sur des vecteurs, des matrices ou des champs tensoriels de façon naturelle. i et j représentent ici les indices des éléments à additionner des matrices A et B. Le résultat est stocké à la même place dans la matrice C. i et j sont calculés en fonction de l'identifiant du thread courant, mais également de l'identifiant du bloc contenant ce thread : on va en effet assigner à chaque bloc le traitement d'une partie déterminée des matrices, ce

qui nécessite pour chaque thread de savoir dans quel bloc il s'exécute. L'invocation du kernel se fait dans le reste du programme C de la façon suivante :

```
int main()
{
  // Taille des matrices
  int N = 4096,

  // Déclaration de matrices A, B et C
  float A[N][N];
  float B[N][N];
  float C[N][N];

  // Remplissage des matrices A et B
  /*  ...  */

  // Invocation du kernel
  dim3 dimBlock(16, 16);
  dim3 dimGrid( (N + dimBlock.x - 1) / dimBlock.x,
                (N + dimBlock.y - 1) / dimBlock.y);
  add_matrix<<<dimGrid, dimBlock>>>(A, B, C);
}
```

Chapitre III

Les représentations parcimonieuses

III.1 Introduction à la parcimonie

Les représentations parcimonieuses sont introduites dans le domaine de détection de mouvement et de suivi d'objets par Mei Ling [Mei 11]. Dans son travail, le suivi d'objets est formulé en tant que problème de représentation parcimonieuse dans le cadre d'un filtre à particules, dans lequel chacun des objets candidats est faiblement représenté par un ensemble de modèles obtenus dans la scène initiale. Pour faire face aux problèmes de l'occlusion, un ensemble de modèles triviaux seront représentés par des matrices d'identification. Motivé par [Mei 11], les représentations parcimonieuses sont alors proposées dans les travaux de [Wang 12] et [Jia 12] pour résoudre la difficulté d'occlusion partielle. Selon [Zhang 12], un système de régularisation est utilisé pour assurer que les objets candidats doivent partager les mêmes coefficients de parcimonie (ou vecteurs de codage). Malheureusement, les méthodes de suivi basées sur les représentations parcimonieuses souffrent du coût énorme de calcul. Pour diminuer ce coût, Bao et al. [Bao 12] propose une approximation accélérée basée sur le gradient. Comme indiqué dans leur résultats, le système de suivi peut être en temps réel. Toutefois, le coût de calcul est sensible au paramètre de réglage.

Comme nous l'avons précédemment évoqué, la recherche de la parcimonie est utile à de nombreux domaines à des fins d'analyse, de modélisation ou encore d'identification. En effet, il est souvent pratique dans le domaine du traitement du signal de représenter l'information dans un autre espace plus propice à l'analyse ou aux manipulations diverses. Classiquement, les signaux sont décomposés dans une base de l'espace sur laquelle la décomposition est unique[Martin 10].

De manière plus générale, on définit un vecteur comme étant parcimonieux si la majorité de ses coefficients sont nuls. Ou plus exactement, qu'un ensemble de signaux de dimension n de $\mathbb{R}^n$ est k-parcimonieux dans une base orthogonale de dimension $n >> k$, si on peut représenter avec une bonne approximation, un quelconque de ces signaux, à l'aide d'environ k composantes de cette base. La figure 3.1 est une illustration schématique pour présenter quelques définitions : celle d'un signal compressé, et comparativement, celle d'un signal parcimonieux. Notons bien que pour un signal compressé ou un signal parcimonieux, on peut obtenir des vecteurs de même dimension k, mais ils n'auront pas la même représentativité du signal. On cherchera à construire la représentation parcimonieuse d'un signal à partir de fonctions définies dans un espace redondant, la décomposition la plus parcimonieuse. Cela permet d'obtenir une décomposition originale d'un signal sur un dictionnaire, faisant intervenir le moins d'éléments possibles.

Les représentations parcimonieuses offrent ainsi un degré de liberté supérieur aux transformations usuelles grâce à cette flexibilité inhérente à l'usage du dictionnaire et plus précisément à la grande variété des atomes qui le composent. Notons bien que les représentations parcimonieuses n'offrent qu'une représentation approximative du signal, à la différence des transformations usuelles réversibles. L'enjeu reste néanmoins l'obtention de la solution la plus parcimonieuse, parmi celles ayant la même erreur de reconstruction, quel que soit le contexte de travail. En effet, la convergence vers la dite solution optimale peut s'avérer complexe dans le cas de signaux bruités. Cette problématique a fait l'objet d'études approfondies [Fuchs 04, Donho 06, Tropp 03], notamment dans le cadre de la détection de signal.

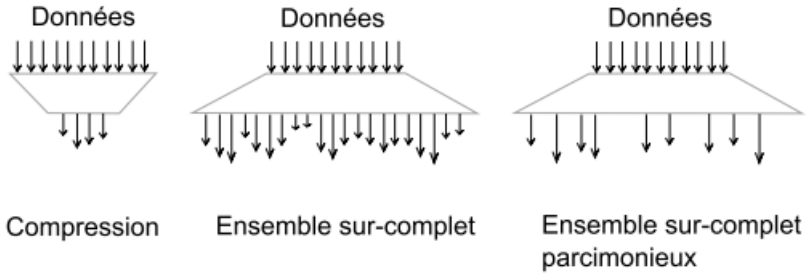


Figure. 3.1 Modélisation de la parcimonie

III.2. Problème à résoudre

III.2.1 Problématique

Trouver la représentation parcimonieuse d'un signal consiste à obtenir la meilleure représentation du signal, c-à-d. la plus parcimonieuse parmi celles ayant la même erreur de reconstruction. Cette représentation est constituée d'un faible nombre d'atomes qui ont été choisis parmi un vaste ensemble de signaux élémentaires, le dictionnaire1 . Générer une représentation parcimonieuse ne correspond pas, à proprement parlé, à une projection sur une base.

Notons y le vecteur représentatif du signal source de dimension m et $A \in \mathbb{R}^{m \times n}$ le dictionnaire, avec $m << n$. Il existe une infinité de solutions quant au choix du vecteur x de dimension n, tel que :

$$y = Ax$$

Le but des représentations parcimonieuses est de trouver parmi l'ensemble des solutions possibles, celles qui sont parcimonieuses i.e. celles pour lesquelles le vecteur x a seulement un faible nombre de coefficients non nuls. Le problème à résoudre est donc le suivant :

$$\rho_0 : \quad \min_x \|x\|_0 \quad sous \quad y = Ax \tag{3.1}$$

où $\|.\|_0$ désigne le nombre de coefficients non nuls.

La minimisation exacte de la norme l0 est un problème NP-complet [Nat95] qui n'a pas de solutions pratiques. Ceci signifie qu'obtenir la solution reviendrait à résoudre un problème combinatoire, i.e. tester toutes les combinaisons d'atomes possibles, méthode bien trop complexe dans un espace de grande dimension et en général, il faudrait de toute façon, m composantes, ce qui est trop dans ce contexte.

On considère donc plutôt le problème suivant : il s'agit de rechercher la solution la plus parcimonieuse tout en tolérant une erreur admissible de reconstruction, notée ε .

$$\widetilde{\rho_0} : \quad \min_x \|x\|_0 \quad sous \quad \|y - Ax\|_2 \leq \varepsilon \tag{3.2}$$

où $\|.\|_2$ est la norme euclidienne l_2: $\|x\|_2 = \sum_{i=1}^{N} \sqrt{|x_i|^2}$

La solution de ce problème d'optimisation est le vecteur parcimonieux x, dont le nombre de coefficients non-nuls est minimal. Le vecteur x conduit à une approximation dont l'erreur de reconstruction est inférieure ou égale à ε. Ce problème est cependant toujours trop difficile à résoudre et on ne le considère jamais.

III.2.2 Distinction de deux axes d'étude

Faire face à la problématique (3.1) soulève donc potentiellement deux difficultés :

– trouver la représentation parcimonieuse pour un dictionnaire A donné ;

– trouver le dictionnaire le mieux adapté au type de signal traité.

La résolution du problème sous-déterminé se fait via des algorithmes que nous décrivons à la section 3.3. Ils tendent à recouvrer une approximation de la solution exacte, la plus acceptable possible : à la fois en terme d'erreur de reconstruction (problème 3.2) et également en terme de parcimonie.

Au delà de la résolution mathématique du critère se pose aussi la question du choix du dictionnaire. On peut dégager à nouveau deux points essentiels. Le premier concerne la redondance du dictionnaire. Quels sont les avantages ? Quels sont les inconvénients ? Nous en discutons au paragraphe 3.4.1. Le second point porte sur la nature des atomes qui composent le dictionnaire. Les atomes doivent nécessairement se prévaloir d'une forte corrélation avec le signal pour espérer le représenter parcimonieusement. Nous présentons au paragraphe 3.4.2 quelques exemples d'atomes caractéristiques, ainsi que leurs spécificités intrinsèques. Etant placé dans un ensemble redondant de représentations possibles, on choisit de ne garder que la plus parcimonieuse d'entre toutes. La parcimonie est directement liée au degré de corrélation des atomes et du signal.

III.3 Algorithmes de décomposition parcimonieuse

Nous décrivons dans cette partie les stratégies algorithmiques développées pour trouver le jeu de coefficients le plus parcimonieux, tout en respectant le critère qui a été choisi. Nous présentons tout d'abord dans la section III.3.1 certains algorithmes itératifs qui visent à obtenir la solution du problème $\widetilde{\rho_0}$. Ces algorithmes ne cherchent pas exactement à résoudre le problème en trouvant l'approximation optimale. Ils raffinent progressivement l'approximation faite du signal par une procédure itérative. Le deuxième type d'algorithmes que nous présentons en section III.3.2 sont des algorithmes de programmation linéaire et quadratique qui visent à résoudre de manière exacte le problème du Basis Pursuit Denoising.

III.3.1 Approche sous-optimale

III.3.1.1 Matching Pursuit

Rappelons ici que nous considérons le problème général de la décomposition d'un signal y

sur un dictionnaire A constitué d'un ensemble de vecteurs unitaires $\{a_\gamma\}_{\gamma\in\tau}$. Le dictionnaire est supposé redondant, ce qui laisse une grande liberté dans le choix du nombre, petit si possible, d'atomes a_γ dont la combinaison linéaire formera une approximation du signal.

Le Matching Pursuit (MP) nommé ainsi par Mallat et Zhang en 1993 [Mallat 93] est un algorithme glouton, connu pour être une alternative à la recherche de la solution optimale.

L'algorithme permet de trouver une approximation sous-optimale. Le principe est donc de sélectionner pas à pas les atomes les plus corrélés avec le signal. Voici décrit ci-après, les étapes du Matching Pursuit.

– La première approximation du signal s'obtient en calculant le projeté orthogonal du signal observé y sur l'atome qui lui est le plus corrélé, notons le $a_{\gamma 1}$. Comme $a_{\gamma 1}$ est normé, la pondération qui minimise la norme du résidu est $\langle y, a_{\gamma 1}\rangle$ et on a donc :

$$R_y^{(1)} = y - \langle y, a_{\gamma 1}\rangle a_{\gamma 1} \tag{3.3}$$

– Lors des itérations suivantes, on recherche l'atome le plus corrélé au résidu courant et on projette ce résidu sur l'atome sélectionné, noté $a_{\gamma k}$. L'estimation obtenue par l'ajout de ce nouvel atome est ajoutée à l'estimation courante du résidu, pour former l'estimation courante du signal y. D'un point de vue plus formel, notons $R_y^{(k)}$ la valeur du résidu au pas

k et b $\hat{y}^{(k)}$ l'approximation courante de y qui s'écrit alors :

$$\hat{y}^{(k)} = \hat{y}^{(k-1)} + \langle R_y^{(k-1)}, a_{\gamma k}\rangle a_{\gamma k} \tag{3.4}$$

Dans la procédure algorithmique que nous venons de décrire, rien n'exclut le fait qu'un atome puisse être sélectionné plusieurs fois. Le nombre d'itérations k n'est donc pas nécessairement égal au nombre total d'atomes a_γ sélectionnés pour la représentation x. L'intérêt majeur du Matching Pursuit est sa grande simplicité d'implémentation et sa relative rapidité d'exécution Contrairement à d'autres algorithmes que nous présentons par la suite, il ne nécessite aucune inversion matricielle. Cependant, cette simplicité a un inconvénient : il peut falloir un grand nombre d'itérations pour converger vers une solution, fait d'autant plus vrai que dans certains cas un atome déjà sélectionné peut à nouveau l'être.

III.3.1.2 Orthogonal matching pursuit

L'Orthogonal Matching Pursuit (OMP) se base sur le même principe que le MP : sélectionner pas à pas les atomes les plus corrélés au signal pour tendre vers une approximation de la solution au problème $\widetilde{\rho_0}$.

Algorithme 3.1 Algorithme du *Matching Pursuit*

ENTRÉES : Le signal source y, le dictionnaire A et le seuil ε

INITIALISATION : $\hat{y}^{(0)} = 0$, $R_y^{(0)} = y \; et \; k = 1$

Tant que $\left\| R_y^{(k-1)} \right\|^2 \geq \varepsilon$ faire

–Recherche de l'atome le plus corrélé $\gamma_k = \text{argmax}_\gamma \left| \langle R_y^{(k-1)}, a_{\gamma k} \rangle \right|$

– Calcul du nouveau coefficient: $x_{\gamma k} = \langle R_y^{(k-1)}, a_{\gamma k} \rangle$

– Mise à jour des données :

1. de l'estimée : $\hat{y}^{(k)} = \hat{y}^{(k-1)} + x_{\gamma k} a_{\gamma k}$
2. du résidu : $R_y^{(k)} = y - \hat{y}^{(k)} = R_y^{(k-1)} - x_{\gamma k} a_{\gamma k}$
3. du vecteur parcimonieux: $x[\gamma k] \leftarrow x_{\gamma k}$

Fin Tant que

La différence réside dans la mise à jour des coefficients. L'objectif de l'OMP est de pallier la faille du MP, qui, comme nous venons de le voir, n'empêche pas la sélection multiple d'un même atome.

Afin d'y parvenir, l'OMP recalcule à chaque pas de l'algorithme la valeur de l'estimée $\hat{y}$. Le MP se base sur la mise à jour du résidu : à chaque itération, on retire au résidu une contribution du seul nouvel atome sélectionné. Pour l'OMP, c'est différent : on évalue la reconstruction courante $\hat{y}^{(k)}$ à chaque fois, ce qui signifie que l'on recalcule tous les coefficients jusqu'alors sélectionnés. L'entrée d'un nouvel atome dans la décomposition modifie l'espace engendré. Il est donc plus judicieux de projeter le signal y, non plus sur le seul nouvel atome, mais sur l'ensemble formé des atomes passés auquel s'ajoute le nouvel atome sélectionné.

Cette mise à jour de tous les coefficients pour chaque nouvel atome choisi pourrait se faire avec la procédure d'orthogonalisation de Gram-Schmidt. On génère donc avec cet algorithme une base orthogonale dont la dimension croît à chaque nouvelle sélection d'un atome. L'ensemble des atomes retenus formant une famille libre de l'espace, on exclut de fait la sélection d'un atome ayant déjà été ajouté à cette base.

Reprenons les mêmes notations que précédemment tout en introduisant deux nouvelles

variables :

– A_k la matrice contenant l'ensemble des atomes sélectionnés à l'itération k :

$A_k = [a_{\gamma 1} \dots a_{\gamma k}]$

– x_k le vecteur contenant uniquement les coefficients non-nuls qui ont été calculés jusqu'au pas courant $x_k = [x_{\gamma 1} \dots x_{\gamma k}]$.

La projection du signal sur l'ensemble formé des atomes sélectionnés jusqu'au pas courant se fait via le calcul de la pseudo-inverse A_k^+. Comme nous l'avons précédemment remarqué, cette méthodologie empêche la sélection d'un atome qui aurait déjà été sélectionné. Cela a l'avantage de surpasser le Matching Pursuit en terme de nombre d'itérations mais nécessite une inversion matricielle à chaque itération. Il existe toutefois des simplifications à l'algorithme présenté qui consistent à déterminer la valeur de A_k^+de manière récursive, i.e. en utilisant la valeur de A_{k-1}^+

. **Algorithme 3.1** Algorithme de *l'Orthogonal Matching Pursuit*

ENTRÉES : Le signal source y, le dictionnaire A et le seuil ε

INITIALISATION : $\hat{y}^{(0)} = 0$, $R_y^{(0)} = y$, $A_0 = [.]$ et $k = 1$

Tant que $\left\| R_y^{(k-1)} \right\|^2 \geq \varepsilon$ faire

–Recherche de l'atome le plus corrélé $\gamma_k = \text{argmax}_\gamma \left| \langle R_y^{(k-1)}, a_{\gamma k} \rangle \right|$

– Ajout du nouvel atome au sous-dictionnaire $A_k = [A_{-1} + a_{\gamma k}]$

– Calcul des coefficients : $x_k = A_k^+ y$ avec $A_k^+ = (A_k^T A_k)^{-1} A_k^T$

– Mise à jour :

1. de l'estimée : $\hat{y}^{(k)} = A_k x_k$
2. du résidu : $R_k = y - \hat{y}^{(k)}$

Fin Tant que

III.3.2 Approche globale

III.3.2.1 Basis Pursuit

Comme la vraie parcimonie ρ_0 n'est pas utilisable en pratique, on utilise la norme l_1 pour contraindre la parcimonie. La norme l_1 est définie de la manière suivante :

$$\|x\|_1 = \sum_{i=1}^{N} |x_i|$$

Cette norme est connue pour favoriser un grand nombre de coefficients nuls. Le problème se formule alors de la manière suivante :

$$\rho_1 : \quad \min_x \|x\|_1 \ sous\ y = Ax \tag{3.5}$$

Des travaux [DH01, Don04] ont montré que dans la majorité des cas, la minimisation de la norme l_1 permet effectivement d'obtenir la représentation la plus parcimonieuse. Ses propriétés de parcimonie n'étant toutefois pas égales à celles que l'on obtiendrait avec l_0. Chen, Donoho et Saunders [CDS98] ont donné le nom de Basis Pursuit (BP) au problème ρ_1 . C'est un problème d'optimisation convexe qui peut être reformulé [CDS98] sous la forme d'un programme linéaire [Dan63, GMW91].

III.3.2.2 Basis Pursuit Denoising

La contrainte égalité du problème (3.5) est trop forte si on veut obtenir une représentation parcimonieuse. Il est par ailleurs réaliste de supposer que les données traitées sont entachées d'un bruit perturbateur. On suppose donc que les données d'observations sont de la forme :

$$y = b + \epsilon$$

où ϵ est un bruit gaussien, b les données sources inconnues et y le signal bruité observé. Voici deux critères qui ont été proposés dans la littérature pour trouver une approximation parcimonieuse $\hat{x}$:

$$\min_x \frac{1}{2}\|y - Ax\|_2^2 \ sous\ \|x\|_1 \leq t \tag{3.6}$$

$$\min_x \|x\|_1 \ sous\ \|y - Ax\|_2^2 < \varepsilon \tag{3.7}$$

Le critère (3.6) fut introduit en 1996 par Tibshirani [DET96], approche dénommée LASSO pour Least Absolute Shrinkage and Selection Operator. La contrainte porte ici sur le nombre maximal de coefficients admis pour la représentation parcimonieuse x.

– Le second critère (3.7) est connu sous le nom de Basis Pursuit Denoising [Fuc97, Fuc98, CDS98]. A l'inverse du LASSO, on fixe l'erreur de reconstruction admissible et non le nombre maximal de coefficients. Il existe une autre formulation du Basis Pursuit Denoising :

$$\widetilde{\rho_1} : \quad \min_x \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1 \tag{3,8}$$

Le paramètre λ sert à ajuster le poids que l'on souhaite donner à chacun des deux éléments de la somme. Il permet d'établir un compromis entre la valeur de l'erreur de reconstruction (premier terme) et le nombre de coefficients non-nuls (deuxième terme). Plus la valeur de λ est grande, plus on privilégie la parcimonie au dépend de la qualité de la reconstruction. Inversement, si λ est très petit, l'approximation obtenue sera de bonne qualité mais peu parcimonieuse.

III.3.2.3 Algorithme du LARS

L'algorithme du LARS, Least Angle Regression a été utilisé et modifié par [EHJT04] pour résoudre le critère du LASSO (3.4). Nous avons vu précédemment que ce critère est une méthode d'estimation par les moindres carrés plus un terme de pénalisation, portant sur le nombre de coefficients non-nuls au sein du vecteur x.

L'algorithme recherche l'estimée itérativement en sélectionnant les atomes les plus corrélés au signal. A l'identique du Matching Pursuit ou de l'Orthogonal Matching Pursuit, à l'itération k, on raffine l'estimation du signal en ajoutant l'apport d'un nouvel atome fortement corrélé au résidu courant, $R_y^{(k)}$. En revanche, la mise à jour du signal estimé $\hat{y}^{(k)}$ se fait par le biais d'un critère géométrique.

Supposons que deux atomes ont déjà été sélectionnés. La mise à jour de b $\hat{y}^{(2)}$se fera le long de la bissectrice formée par les deux atomes sélectionnés. Ainsi au pas k +1, si k atomes ont été retenus, l'estimée est mise à jour le long de la direction équiangulaire formée par les k atomes retenus. Notons $d^{(k+1)}$ cette direction ; l'expression récursive de l'estimée est donc la suivante :

$$\hat{y}^{(k+1)} = \hat{y}^{(k)} + \gamma^{(k+1)} + d^{(k+1)}$$

avec $\gamma^{(k+1)}$ un paramètre qui règle la contribution du nouvel atome.

A chaque itération, on choisit un nouvel atome de telle sorte que sa corrélation avec le résidu courant soit égale aux corrélations des précédents atomes sélectionnés avec le résidu courant. Pour atteindre cette contrainte, on calcule le pas γ qui permet de déterminer de combien l'estimée doit se décaler dans la direction d pour que le nouvel atome soit autant corrélé avec le résidu courant que les atomes déjà sélectionnés.

III.4 Dictionnaires

III.4.1 Des bases orthonormales aux dictionnaires redondants

Le signal peut se représenter d'une infinité de manières via les colonnes du dictionnaire, les atomes a_γ . Le signal est décomposé sous la forme d'une combinaison linéaire pondérée de fonctions élémentaires :

$$y = \sum_{\gamma \in \tau} x_\gamma a_\gamma \qquad (3,9)$$

où x_γ sont les coefficients de pondération des atomes a_γ et τ l'ensemble fini d'indices représentant les fonctions élémentaires choisies.

III.4.1.1 Enrichir le dictionnaire

La nouveauté ces dernières années a été d'élargir le spectre des fonctions élémentaires, en se détachant des bases orthonormales usuelles [DH01, DE03, Fuc02, GN02, Dau04]. Le dictionnaire se compose alors de diverses fonctions, de natures différentes. Cette diversité est à l'origine de la notion de redondance. En effet dans le cas classique d'une base orthonormale, la décomposition obtenue est unique puisque les vecteurs de base sont linéairement indépendants. En revanche, lorsqu'on enrichit le dictionnaire avec un surplus d'atomes, on s'expose au fait que le signal aura de multiples représentations possibles.

III.4.1.2 Danger de la redondance

Il existe réellement un bémol quant à la recherche du dictionnaire sur-complet. Choisir un ensemble exhaustif de fonctions peut devenir un inconvénient si ces atomes ont une trop faible corrélation avec le signal. Dans le cas de la concaténation de deux bases orthonormales (ou non), l'intelligence repose dans le degré de complémentarité des dictionnaires concaténés.

Il est judicieux de choisir comme deuxième dictionnaire un ensemble d'atomes qui permettra de reconstituer certaines des caractéristiques du signal, que l'autre dictionnaire ne sera pas en mesure de représenter parcimonieusement.

III.4.1.3 Atomes corrélés au signal

Le choix des atomes dépend de l'application pour laquelle on destine l'utilisation des représentations parcimonieuses. Il s'est avéré utile, voire indispensable, d'avoir des atomes ayant les mêmes caractéristiques que le signal source que l'on cherche à modéliser. Prenons le cas des images, la modélisation d'un contour sera évidente si les atomes eux-mêmes possèdent une structure visuellement proche d'un contour. De même, se pose la question du facteur d'échelle :
comment représenter efficacement i.e. trouver la représentation la plus parcimonieuse, d'un motif ayant cinq fois la taille des atomes ? Ainsi, plus le dictionnaire possède de fonctions structurelles variées, permettant la représentation des signaux à différentes résolutions, plus les chances d'obtenir une représentation parcimonieuse sont fortes.

III.4.1.4 Dictionnaires adaptatifs

Au-delà de l'élaboration de tels dictionnaires, il existe également les méthodes d'apprentissage[PFS07]. Le dictionnaire est adaptatif car construit à partir des données sources elles-mêmes. L'avantage est bien sûr la grande liberté qu'offre l'adaptabilité pour représenter n'importe quel signal. Si l'on ne se pose plus la question du choix du type d'atomes, il se pose en revanche celle de la complexité de telles méthodes, qui, à l'heure actuelle, sont encore très lourdes.

Chapitre IV

Détection et estimation de mouvement: Méthodes proposées et contribution

IV.I Introduction

Le but de cette thèse est la détection et l'estimation de mouvement dans une séquence vidéo. Nous nous sommes intéressés précisément à la détection de personnes dans des scènes encombrées. Plusieurs approches ont été testées pour la détection dans ce travail, on compte: la détection des personnes via des systèmes de classification (Haar et Hog), par des caractéristiques robustes et par la soustraction du fond. Chacune de ces méthodes a des avantages et des inconvénients.

Pour l'estimation du mouvement, nous jugeons très nécessaire l'étape de reconnaissance et de discrimination des objets pour estimer le vecteur mouvement de chacun d'eux. Pour cela, il est préférable d'utiliser une approche discriminative basée sur les représentations parcimonieuses au lieu d'utiliser une approche générative basée sur le calcul de la ressemblance des apparence. Nous avons proposé l'algorithme Orthogonal Matching Pursuit (OMP), pour l'obtention du vecteur de parcimonie qui sert à classifier chaque objet. Cet algorithme utilise un dictionnaire contenant les caractéristiques de chaque objet détecté. Une fois cet objet classifié, le dictionnaire est mis à jour. le choix de l'algorithme OMP est du pour sa rapidité de convergence et sa précision. Pour accélérer le traitement, l'algorithme OMP a été implémenté en parallèle. Nous présenterons aussi dans ce chapitre, le nouveau algorithme en parallèle en décrivant les étapes essentielles, et surtout en comparant les résultats obtenus avec ceux de l'implémentation séquentielle. Nous décrivons aussi la manière avec laquelle chaque objet est représenté. la représentation des objets est basée sur deux caractéristiques:

- Caractéristique basée sur l'apparence
- Caractéristique basée sur la position dans le plan 2D

Une nouvelle approche de représentation de la position est utilisée dans ce travail permettant d'augmenter la précision et surtout permettant une adaptabilité avec les représentations parcimonieuses.

IV.2 Méthode proposée pour la détection des objets:

Le suivi d'objet est un sujet très intéressant dans le domaine du multimédia et de la vision par ordinateur, Il touche en particulier la surveillance vidéo qui est très utilisée dans nos jour par tous les gouvernements. Le but du suivi d'objets mobiles appelé "Tracking" est de déterminer l'emplacement de chaque objet dans la scène en dépit des différents problèmes tel que les variations de la lumière et de l'échelle, l'occlusion complète ou partielle et d'autres problèmes qui rendent cette tache difficile. Pour résoudre ces problèmes et créer un système assez fiable pour suivre un objet dans une scène, plusieurs chercheurs proposent des solutions.

Dans la partie suivante nous présenterons nos expérimentations pour la détection des objets. Nous précisons que certaines approches ont été testées dans différents types de scènes. Cependant, nous avons choisi la soustraction du fond après avoir testé d'autre méthodes, car elles ne s'avèrent pas très efficaces dans le cas des scènes encombrées et acquises de loin. On détaillera les résultats dans cette section.

IV.2.1. Détection par classification

Un histogramme de gradient orienté (HOG) est une caractéristique utilisée en vision par ordinateur pour la détection d'objet. La technique calcule des histogrammes locaux de l'orientation du gradient sur une grille dense, c'est-à-dire sur des zones régulièrement réparties sur l'image. Elle possède des points communs avec les SIFT, les Shape contextes et les histogrammes d'orientation de contours, mais s'en diffère notamment par l'utilisation d'une grille dense. La méthode s'est montrée particulièrement efficace pour la détection de personnes. Sauf que dans certain cas, comme montré dans la Figure 4.2, cette méthode risque de ne pas être précise.

Figure.4.1. Image du benchmark utilisé PET'S 09

Après avoir testé les méthode de détection mentionné dans les points précédents on constate que les approches non paramétrique comme l'histogramme de gradient orienté (HOG) ainsi

que la détection de mouvement basée sur la cohérence ne donne pas de très bon résultats dans les scènes encombrées.

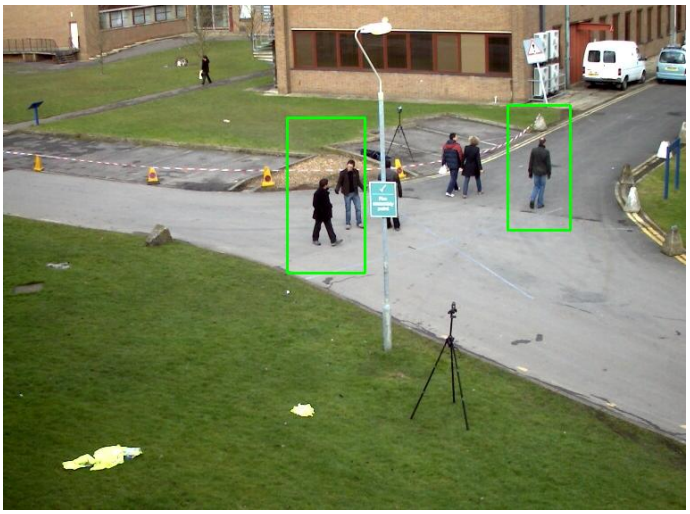


Figure 4.2 : Exemple de détection de personne avec la méthode HOG

Le système de classification Haar basé sur un dictionnaire d'apprentissage, lui aussi peut donner des résultats pas assez précis. On remarque dans la Figure 4.3 que des objets appartenant au fond sont détectés comme personnes mobiles (un poteau, une fenêtre..) , alors que c'est faux. En plus même si les objets sont détectés, ils sont délimités par des zones largement supérieures, ceci peut causer de grave conséquence lors de l'étape de la représentation des objets, car une bonne partie du fond sera comptabilisée lors de l'extraction des caractéristiques nécessaires à la représentation de l'objet, ces caractéristiques peuvent être constituées de l'histogramme (RGB, HS ou RGI (red-green-intensity)), caractéristiques texture LBP (local binary patterns). Dans tous les cas, les informations appartenant au fond peuvent gravement fausser la représentation de l'objet nécessaire pour son suivi. Dans [Breitenstein 11] les auteurs ont utilisé le classificateur Hog comme moyen de détection pour la création de leur système de suivi, cependant, l'extraction des caractéristiques de la zone rectangulaire contenant beaucoup d'information du fond, a rendu leur manière de décrire les objets pas assez fiable pour un suivi correct et une bonne reconnaissance.

IV.2.2. Détection par des Caractéristiques photométriques

Nous avons aussi testé la de détection des objets de l'avant plan par des caractéristiques photométriques, car le mouvement n’est pas toujours suffisant pour différencier les objets du fond.

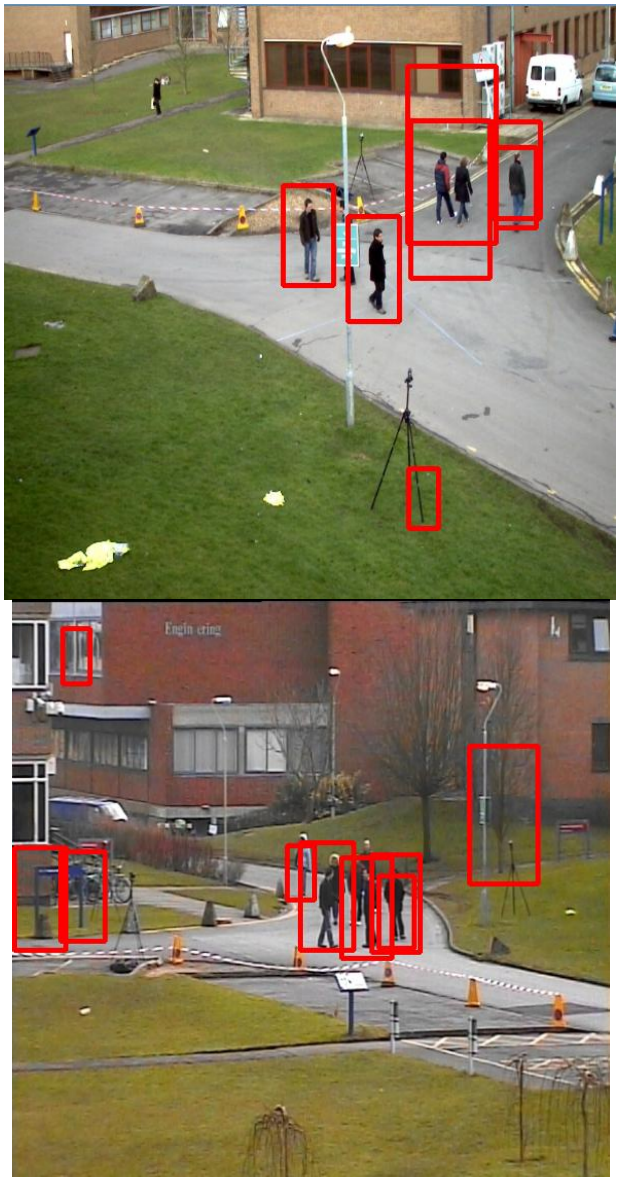


Figure 4.3 : Exemple de détection de personne avec la méthode Haar

Des caractéristiques photométriques doivent donc être ajoutées au descripteur. Le principe est de procéder à l'extraction des point d'intérêt dans une image. Ces points d'intérêts doivent être robustes aux changements de la lumière. Pour détecter l'avant plan de l'arrière plan on a besoin des caractéristiques de deux images consécutives. On note par F^t l'ensemble des points d'intérêts de l'image t, et par F^{t-1} l'ensemble des points d'intérêts appartenant à l'image t-1. Dans chaque ensemble F, chaque point d'intérêt $P(x, y)$ est représenté par son descripteur $\hat{D}(p_{x,y})$ et ses coordonnées $\hat{C}(p_{x,y})$.

L'ensemble des points d'intérêts appartenant à l'arrière plan φ est défini comme suit:

$$P_t \in \varphi si\ \hat{D}(p^t) = \hat{D}(p^{t-1})\ et \hat{C}(p^t) = \hat{C}(p^{t-1}) \qquad (4.1)$$

$\bar{\varphi}$ représente l'ensemble des points d'intérêts appartenant à l'avant plan:

$$P_t \in \bar{\varphi}\ si\ \hat{D}(p^t) = \hat{D}(p^{t-1})\ et \hat{C}(p^t) \neq \hat{C}(p^{t-1}) \qquad (4.2)$$

On fait correspondre les caractéristiques des deux images, si le point n'a pas changé de coordonnées il est jugé comme appartenant au fond (formule 4.1). Si le point change de position alors il fait partie d'une zone appartenant à l'avant plan(formule 4.2). Dans la Figure 4.4.A, on remarque des cercles en rouge qui représentent des points d'intérêt faisant partie des objets en mouvement. Chaque caractéristique robuste accélérée (SURF) [Bay 08] garde le même descripteur et la même orientation dans chaque image, sauf les coordonnées qui changent dans le cas où la zone contenant cette caractéristique change d'emplacement dans la scène.

(A)

(B)

Figure 4.4 Exemple de détection par des Caractéristiques photométriques

Pour la figure 4.4.B, les cercles en vert représentent tous les points d'intérêt appartenant à la scène (l'avant plan et l'arrière plan).

Cette approche essayée dans notre travail semble performante et donne de bons résultats pour la détection du mouvement; sauf que le temps d'exécution est très élevé.

IV.2.3. Détection par la différence d'images

Dans le cas d'une caméra fixe, l'utilisation de la soustraction des images est un moyen efficace pour la détection de mouvement. Le résultat de cette opération peut cependant contenir un peu de bruit indésirable en raison du changement de fond tel que les changements d'illumination et les mouvements indésirables qui doivent être filtrés. Nous avons implémenté deux approches dans ce travail faisant partie de cette catégorie: soustraction de deux images consécutives et soustraction du fond.

IV.2.3.1 Soustraction de deux images consécutives

Cette opération consiste à soustraire une image acquise à l'instant *t* d'une autre acquise à l'instant *t+1*. Le résultat de cette opération est représenté par un masque binaire *R* tel que

$$R(x,y) = \begin{cases} 1, & si |I_t(x,y) - I_{t+1}(x,y)| \geq \tau \\ 0 & sinon \end{cases} \qquad (4,3)$$

où τ représente un seuil.

L'inconvénient de cette méthode est que si l'objet même appartenant à l'avant plan ne bouge pas pendant quelque instant, il sera considéré comme appartenant au fond, et si son mouvement n'est pas très important, la région du masque résultant ne pourra pas déterminer la forme exacte de cette objet. La figure 4.5 est un exemple de cette opération. On remarque clairement que la forme des objets après la détection est complètement déformée. En plus on remarque aussi qu'un objet (une personne) n'est pas détecté, ceci est du au fait que cet objet n'a pas bougé entre l'instant *t* et *t+1*.

L'utilisation du mélange gaussien peut être très utile pour la modélisation de fond. La méthode Mog (mixture of gaussian) se base sur un système d'apprentissage qui a comme but de séparer les pixels de l'avant plan. La figure 4.6 est un exemple de détection basée sur la méthode Mog, avec un paramètre de mixture = 3 et un taux d'historique =6.

Figure 4.5 Résultat de la méthode de la soustraction de deux images consécutives

Figure 4.6 Résultat de la méthode Mog

IV.2.3.2 Soustraction de fond

Dans ce travail nous nous basons principalement sur la soustraction de fond pour la détection de mouvement et ceci pour sa simplicité d'implémentation et sa rapidité d'exécution. Cette méthode a le même principe que la méthode décrite précédemment, sauf que la soustraction se fait entre une image acquise à l'instant t est un model de fond statique. L'inconvénient de cette approche est la création du model de fond. Si l'environnement de la scène observée change, alors il faut mettre à jour le modèle de fond; ces événements peuvent être par exemple, la variation du temps d'acquisition (matin ou soir), des nuages qui peuvent éclipser la source de lumière ou d'autre paramètres tel que le mouvement de la caméra ou le changement d'emplacement d'objet sensé être immobile. Malgré ces inconvénients la soustraction de fond reste une approche très utilisée pour la détection de mouvement dans différent travaux de l'état de l'art.

La Figure.4.7 illustre l'opération de la soustraction de fond, elle consiste à la différence absolue entre le fond (Figure.4.7.a) et une autre scène acquise à un moment donné (Figure.4.7.b). Après seuillage du résultat de la soustraction de fond, les formes des corps sont visibles sur la figure 4.7.c.

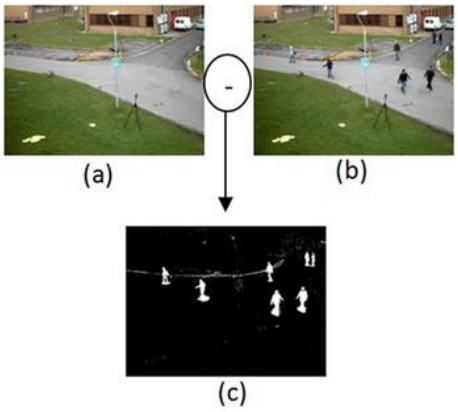
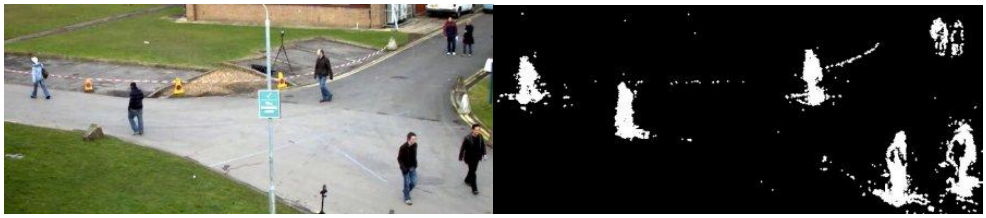


Figure.4.7 Résultat de la soustraction de fond.(a) Fond référence. (b) scène observée. (c) Résultat de la soustraction de fond seuillé.

On remarque que sur la figure. 4.8.b, le résultat de la soustraction ne comprend pas seulement les formes "les corps", mais le résultat contient aussi du bruit. Cette opération reste très dépendante du seuil utilisé, et le choix de ce dernier reste très critique.

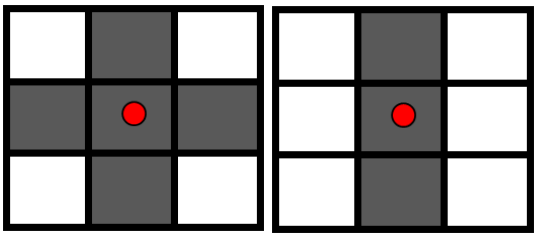


(a) (b)

Figure 4.8 Agrandissement du résultats

Pour améliorer l'extraction des corps des objets mobiles détectés, des opérations de morphologie mathématique sont utilisées. Une ouverture est appliquée sur le masque binaire résultant de la soustraction de fond pour améliorer les résultats et bien délimiter les formes des objets en mouvement. On présentera dans ce qui suit une brève description de ces opérations morphologiques ainsi que leur application dans notre travail.

Elément structurant: un élément structurant est un masque binaire constitué de pixels noirs et blancs, le masque est muni d'un point d'ancrage. La figure 4.9 représente un exemple de deux éléments structurants, le point rouge au milieu désigne le point d'ancrage.



(a) (b)

Figure 4.9: Elément structurant

Soit (x, y) les coordonnées d'une image binaire X et M un élément structurant. $M_{(x,y)}$ représente l'ensemble des pixels noirs de X qui coïncident avec les pixels noir de M lorsque le point d'ancrage est superposé au pixel de coordonnée(x, y).

L'érosion: Soit une image binaire X et un élément structurant M, l'érosion de X par M est une image binaire définie par:

$$Ero_M(X) = \{(x, y) | M_{x,y} \subset X\} \quad (4,4)$$

La dilatation: Soit une image binaire X et un élément structurant M, la dilatation de X par M est une image binaire définie par:

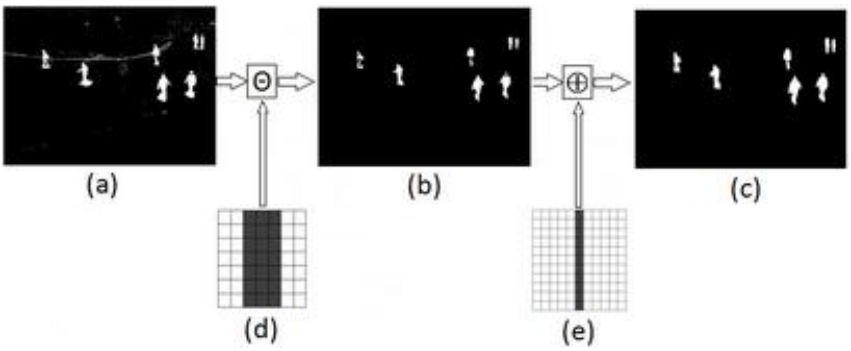
$$Dil_M(X) = \{(x, y) | M_{x,y} \cap X \neq \emptyset\} \quad (4,5)$$

L'érosion est appliquée pour éliminer le bruit tel que celui caractérisé par la ligne blanche due au mouvement du ruban horizontal. La figure 4.10 représente le résultat de l'application de l'érosion avec un élément structurant d'une forme ressemblante à celle de la figure 4.9.a sur une image binaire résultante de l'opération de soustraction de fond.

(a) (b) (c)

Figure 4.10 Résultat de l'érosion. (a) le masque binaire originale. (b) Résultat de l'érosion avec un élément structurant de (3x3). (b) Résultat de l'érosion avec un élément structurant de (5x5).

On remarque clairement sur la figure 4.10.c que le bruit a disparu, cependant on remarque aussi que cette opération a généré un effet secondaire, les parties constituants le corps sont dissociés. Pour remédier à ce problème, nous proposons une ouverture c-à-d une érosion suivie par une dilatation. Pour l'érosion, un élément structurant constitué d'une matrice (7x7). L'élément structurant proposée élimine les pixels du bruit ou ceux constituant des formes horizontales. Cependant, l'érosion génère un effet indésirable et dissocie les membres d'organes comme on peut le voir sur la Fig. 4.10.c. Nous appliquons une dilatation avec un autre élément structurant composé d'une (9x9) matrice pour reconstruire les formes des corps. On peut observer maintenant sur la figure 4.11.c que le résultat de l'ouverture améliore nettement la reconstitution de la forme des objets détectés.



Figue. 4.11. Filtrage du résultat. (a) Résultat de la soustraction de fond. (b) Résultat de l'érosion. (c) Résultat de dilatation. (d) Elément structurant utilisé pour l'érosion. (e) Elément structurant utilisé pour la dilatation

IV.3 Méthode proposée pour la représentation des objets

La manière avec laquelle on représente un objet a une très grande influence pour le processus d'indexation et de reconnaissance des objets dans des scènes encombrées. Nous avons précisé précédemment que nous nous intéressons dans notre travail à la détection et le suivi de personnes en mouvement dans des scènes encombrées. Pour bien estimer le vecteur mouvement de chaque objet, nous proposons pour sa reconnaissance de le représenter avec deux caractéristiques fondamentales: son apparence couleur, et sa position dans la scène.

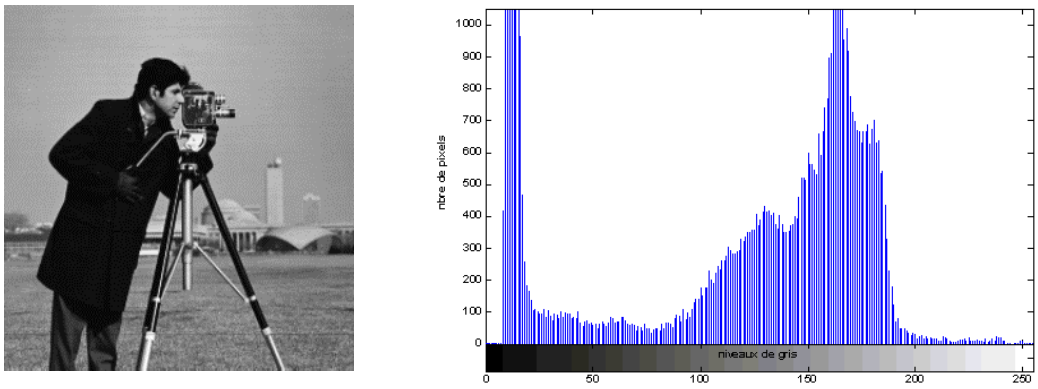
Pour les caractéristiques couleurs, chaque objet est représenté par deux histogrammes couleurs, l'un caractérisant le haut du corps et l'autre caractérisant le bas du corps par rapport au centre. Concernant les caractéristiques de la position, nous proposons une nouvelle représentation sur le plan 2D adéquate à l'étape de reconnaissance basée sur les représentation parcimonieuse[Elbahri 15].

IV.3.1 La représentation par l'apparence

Dans beaucoup de travaux relatifs de l'état de l'art [Breitenstein 11, Ben 11, Possegger 14, Lu 13], chaque objet détecté est représenté par son histogramme couleur, nous nous sommes basés sur le même principe de représentation. Cependant pour augmenter la précision de la reconnaissance, nous avons modifié la manière d'extraction des caractéristiques couleurs. Avant de présenter notre contribution pour l'extraction des caractéristiques couleurs, nous allons de ce qui suit définir le principe du calcul de l'histogramme et les différentes manières de calcul de la similitude entre deux histogrammes.

IV.3.1.1 Les histogrammes

Pour les images numériques un histogramme est tout simplement une représentation graphique du nombre de pixels de niveau d'intensité de la luminance ou des couleurs. Par exemple, pour un histogramme de luminance normal, le graphe présenté par la figure 4.12.b montre le nombre de pixels pour chaque niveau de gris de l'image représenté par la figure 4.12.a.



(a) (b)

Figure 4.12: représentation graphique d'un histogramme de niveaux de gris

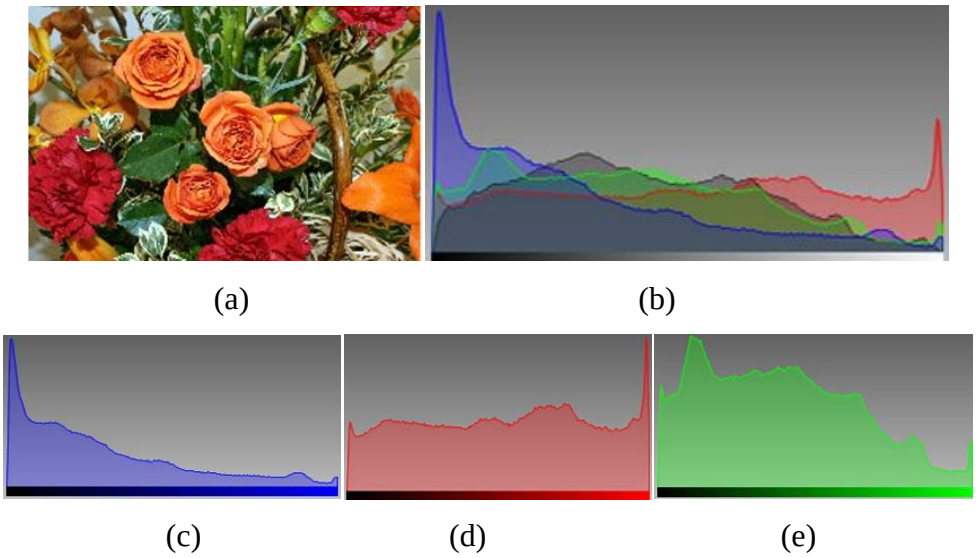
On effectue parfois une quantification, qui regroupe plusieurs valeurs d'intensité en une seule classe, ce qui peut permettre de mieux visualiser la distribution des intensités de l'image.

Les histogrammes sont en général normalisés, en divisant les valeurs de chaque classe par le nombre total de pixels de l'image. La valeur d'une classe varie alors entre 0 et 1, et peut s'interpréter comme la probabilité d'occurrence de la classe dans l'image. L'histogramme peut alors être vu comme une densité de probabilité. Pour une image X en niveaux de gris codée sur L niveaux, on définit n_k le nombre d'occurrences du niveau X_k. La probabilité d'occurrence d'un pixel de niveau X_k dans l'image est:

$$p_x(x_k) = p_x(x = x_k) = \frac{n_k}{n}, 0 \leq k < L \qquad (4,6)$$

Un histogramme de couleur est un histogramme qui détermine tout simplement le niveau de couleur pour chaque canal de couleur RGB séparément. Un histogramme de couleurs est similaire à un histogramme de luminance normale. Cependant, au lieu de déterminer la distribution des pixels du noir au blanc comme avec l'histogramme de luminance, un histogramme de couleurs montre la répartition de la luminosité pour chaque couleur individuellement. Ceci est important parce qu' on peut facilement savoir si une couleur domine la représentation de l'image.

Il ya deux types de base d'histogrammes couleurs. La première catégorie, est l'histogramme RGB qui représente une combinaison des trois couleurs et peut-être même l'histogramme de luminance tous ensemble (voir figure 4.13.b) . Les seconds sont des histogrammes individuels pour chaque couleur distincte.



(a) (b)

(c) (d) (e)

Figure 4.13 Exemple des différents types d'histogramme

(a) Une image exemple (b) l'histogramme des trois couleurs RGB et de la luminance. (c) L'histogramme de la couleur bleu. (d) L'histogramme de la couleur rouge. (e) L'histogramme de la couleur verte.

IV.3.1.2 La mesure de similarité

Dans le présent travail comme d'autres travaux déjà réalisés qui étudient les variations temporelles du contenu visuel dans une vidéo, ou pour trouver des images similaires à une image requête il est utile de calculer une distance (ou plus généralement une mesure de similarité/dissimilarité) entre histogrammes. On définit pour cela un certain nombre de

distances pour mesurer si deux histogrammes sont « proches » l'un de l'autre. On peut mesurer la similarité en calculant la corrélation qui peut être mesurée comme suit:

Soit $h_1 et\ h_2$deux histogrammes de même taille N(i.e. avec le même nombre de classes)

$$d(h_1, h_2) = \frac{\sum_{i=1}^{N} h_1(i) h_2(i)}{\sqrt{\sum_{i=1}^{N} h_1(i)^2 \sum_{i=1}^{N} h_2(i)^2}} \quad (4,7)$$

Ou par l'intersection qui peut être calculée comme suite:

$$d(h_1, h_2) = \sum_{i=1}^{N} \min[h_1(i), h_2(i)] \quad (4,8)$$

IV.3.1.3 Extraction des caractéristiques couleurs

Après l'extraction des formes des objets, la détection des contours est appliquée à l'image binaire résultante de la soustraction de fond pour délimiter la silhouette du corps de l'objet mobile (voir Figure 4.14). Dans l'approche proposée, l'histogramme est dérivé de cette silhouette, tandis que la majorité des techniques utilisées dans ce domaine, l'histogramme est calculé à partir du rectangle délimitant l'objet [Breitenstein 11, Ben 11, Possegger 14, Lu 13].

Figure. 4.14 Détection de contour

Pour chaque objet détecté, un masque rectangulaire est extrait de l'image binaire. La figure 4.15 montre un zoom d'un objet détecté (Fig.4.15.a) ainsi que le masque extrait pour le même objet (Fig.4.15.b). La valeur 0 représente le fond (noir), tandis que la valeur 1 représente les pixels de la silhouette du corps de l'objet (blanc).

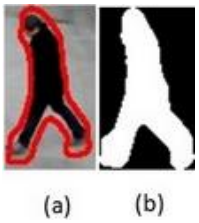


Fig. 4.15 Objet détecté (a) Contour de l'objet (b) Masque correspondant à l'objet.

L'utilisation de silhouettes comme des masques d'objets rend les histogrammes plus précis puisque seuls les pixels représentant les objets en mouvement sont comptabilisés. Dans l'exemple suivant (Fig.4.15), les histogrammes sont calculés à partir de deux images représentant la même personne dans des cadres différents. A partir de chaque rectangle de délimitation, l'histogramme est calculé de deux façons: la première façon utilisant le masque et le seconde sans masque. Après cela, la similitude est mesurée entre chaque paire d'histogrammes. les figures 4.15.a et 4.15.c montrent la même personne dans deux différentes scènes. Les pixels noirs représentent 42% de l'image représentant cette personne(zone rectangulaire), par conséquent, les pixels de fond faussent considérablement la courbe du graphique représentant les différents histogrammes de la même personne détectée dans des images différentes ayant des fonds différents. Par contre la figure.4.15.b montre seulement une personne sans les pixels de fond.

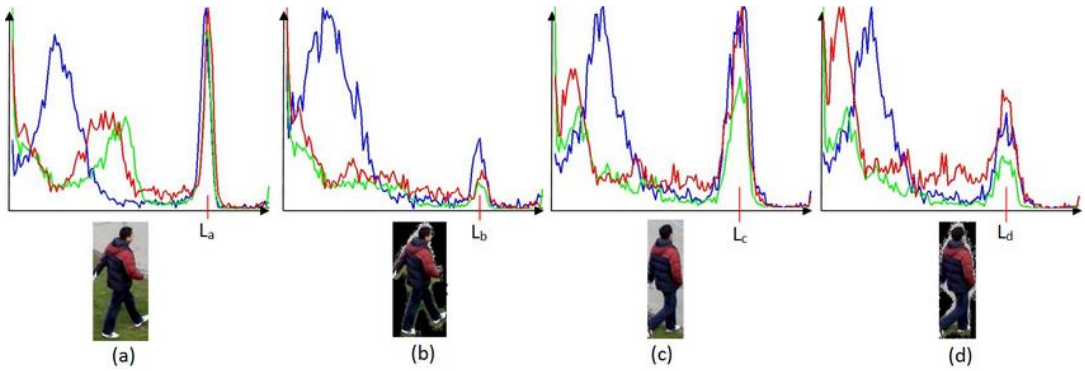


Figure.4.16. Exemple d' histogrammes des images.

On constate que dans le graphe correspondant à Fig.4.15.a, le pic noté L_a représente les pixels de l'arrière-plan et on remarque qu'il est plus élevé que celui au niveau L_a du graphique correspondant à la Fig.4.15.b. Le pic au niveau de L_a est réduit lorsque l'histogramme est calculé en utilisant le masque. L'objectif de cette approche est d'améliorer la similitude entre

deux histogrammes représentant la même personne et en réduisant l'effet négatif de la comptabilisation des pixels de l'arrière-plan. Par exemple, les graphes des spectres verts des figures 4.15.b et 4.15.d sont plus semblables que ceux des figures 4.15.a et 4.15.c. Ceci est confirmé par la similitude mesurée dans les deux cas. Le degré de similarité entre les deux images peut être mesuré à l'aide de la corrélation entre les histogrammes. On note par $h_1 et\ h_2$ les histogrammes de deux images I_1 et I_2 respectivement. On modifie la formule (4,7) pour définir la corrélation entre ces histogrammes et selon les trois couleurs:

$$d_c(h_1^c, h_2^c) = \frac{\sum_{i=1}^{N} h_1^c(i) h_2^c(i)}{\sqrt{\sum_{i=1}^{N} h_1^c(i)^2 \sum_{i=1}^{N} h_2^c(i)^2}} \tag{4.9}$$

où $c \in \{R, G, B\}$ avec$0 \leq d_c(h_1^c, h_2^c) \leq 1$, et si $d_c(h_1^c, h_2^c) \approx 1$ alors h_1 et h_2 sont similaires.

On note par $h_1^c, \hat{h}_1^c, h_2, \hat{h}_2^c$, les histogrammes correspondants aux figures 4.15.a, 4.15.b, 4.15.c et 4.15.d, respectivement. La corrélation moyenne entre les histogrammes des trois canaux de couleur de l'exemple présenté par la figure 4.15 est la suivante: $\overline{d_c(h_1^c, h_2^c)} = 0{,}53\ et\ \overline{d_c(\hat{h}_1^c, \hat{h}_2^c)} = 0{,}83$. On remarque que $\overline{d_c(\hat{h}_1^c, \hat{h}_2^c)} > \overline{d_c(h_1^c, h_2^c)}$.
La corrélation d'histogrammes représentant la même personne dans différentes scènes avec différents milieux utilisant des masques de silhouette est plus grande que celle utilisant une forme rectangulaire pour décrire l'objet. En partant de cette démonstration, nous confirmons que l'utilisation de la silhouette comme un masque dans la construction de l'histogramme est très bénéfique.

IV.3.2 Représentation de l'objet par sa position sur le plan 2D

La position des objets en mouvement contribue à la reconnaissance des objets. Il est représenté dans la littérature avec les coordonnées du centre de l'objet[Lu 13]. L'algorithme de classification utilisé dans ce travail pour la reconnaissance des objets est basé sur le produit intérieur pour la sélection de la colonne la plus corrélée dans le dictionnaire avec l'objet de test. Cependant, le produit interne de simples coordonnées ne peut pas fournir des informations sur la distance entre deux objets: deux objets peuvent être fortement corrélés mais loin les uns des autres et les objets proches peuvent être faiblement corrélés.
Pour illustrer notre méthode, on note par P_i s les positions de trois objets détectés à l'instant t_1 et par $\hat{P}_j$ s les positions des même objets a l'instant t_2, où $P_i, \hat{P}_j \in \mathbb{R}^{2\times 1}$ et $\mathrm{i}, \mathrm{j} \in \{1,2,3\}$. Suivant l'exemple de la figure.4.17. on obtient l'équation suivante:

$$\forall\, i, argmax_j \langle P_i, \hat{P}_j \rangle = 3 \tag{4.10}$$

où $\langle P_i, \hat{P}_j \rangle = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \times \begin{pmatrix} \hat{x}_j \\ \hat{y}_j \end{pmatrix}^T$.

Après avoir observé les résultats du produit intérieur des différentes positions dans le tableau 1, nous confirmons la formule (4.2), et le plus grand produit intérieur est celui généré avec la position $\hat{P}_3$ correspondant à l'objet le plus éloigné de l'origine. Cependant, nous avons besoin de représenter la position de telle sorte que:

$$\forall\, i, argmax_j \langle P_i, \hat{\hat{P}}_j \rangle = i \tag{4.11}$$

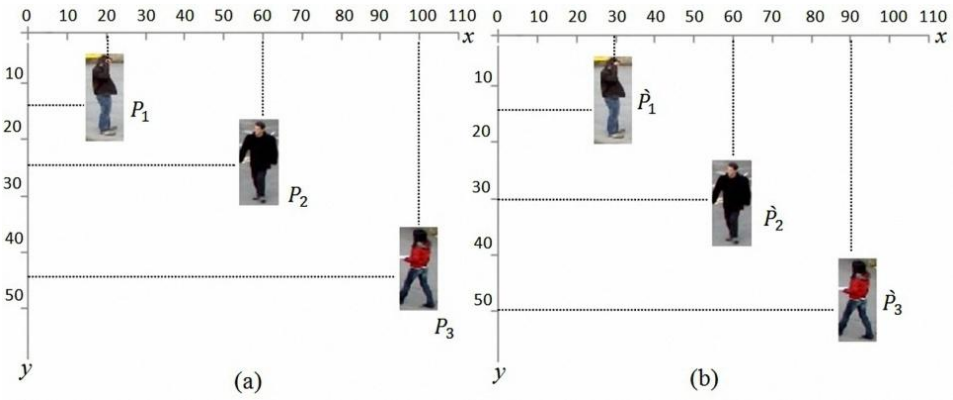


Figure 4.17 Exemple de positions

Pour satisfaire la formule (4.11), une nouvelle représentation de la position est proposée dans ce travail. L'idée est de créer une structure pour coder la position de chaque objet détecté. Cette structure est dérivée d'une matrice $Q \in \mathbb{R}^{m,n}$ crée pour chaque objet détecté. Cette matrice, appelée matrice de position, est utilisée pour échantillonner l'image en petites zones. La figure 4.19 représente les matrices de position des trois objets détectés. Le cadre dans Figure.4.19.a.1 est acquis à t_1 et la matrice de position dans Figure. 4.19.a.2 est utilisée pour représenter la position de la personne que nous appelons "personne1". Les figures 4.19.b.1 et 4.19.c.1 montrent les scènes acquises à t_2, et les matrices de position correspondant à deux personnes ("Personne1" et un autre que nous appelons "personne2") sont illustrées par les figures et 4.19.b.2 4.19.c.2, respectivement. On précise que dans chaque matrice de position, la cellule rouge correspond au centre de l'objet. Pour les autres cellules, leur couleur est modifiée en fonction de la distance qui les sépare de la cellule rouge.

Tableau 1: Résultat du produit intérieur de la position des objets selon la Figure.4.17

$\langle P_i, \hat{P}_j \rangle$	j=1: (30,15)	j=2: (60,30)	j=3: (90,50)
i=1: (20,15)	825	1650	**2250**
i=2: (60,25)	2175	4350	**6650**
i=3:(100,45)	3675	7350	**11250**

Pour implémenter cette approche, la matrice $Q \in \mathbb{R}^{m,n}$ Contient initialement des zéro. On pose la valeur $\alpha \in \mathbb{R}^{+}$ Dans la cellule $Q_{i,j}$ Correspondant aux coordonnés $(x\ , y)$ du centre de l'objet. Où $i = \left\lfloor \frac{x}{\frac{h}{m}} \right\rfloor, j = \left\lfloor \frac{y}{\frac{w}{n}} \right\rfloor$, avec $i, j \in \mathbb{N}$, h est la hauteur de l'image et w sa largeur.

Les autres valeurs des cellules sont définies par:

$$Q_{i',j'} = \begin{cases} \alpha - \gamma\vartheta \ if \geq 0 \\ else\ 0 \end{cases} \tag{4.12}$$

où $\gamma = \max\left(\left|i' - i\right|, \left|j' - j\right|\right) and\ \vartheta \in \mathbb{R}: 0 < \vartheta < \alpha$

On peut dire cette représentation est similaire à une cloche gaussienne en deux dimensions (Figure 4.19), plus on s'éloigne du centre plus la on perd de l'énergie.

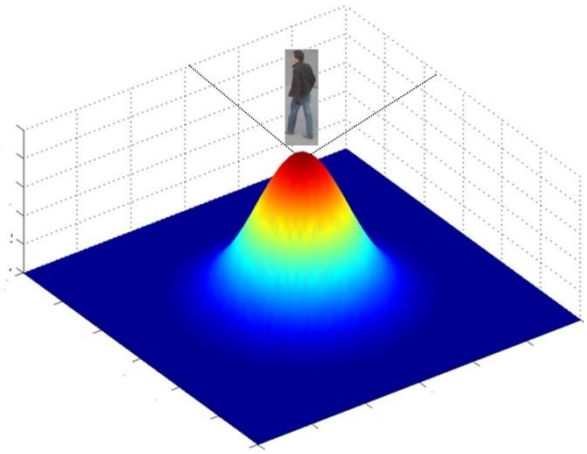


Figure 4.18: Représentation de la position par une fonction gaussienne

Dans un domaine continu, on peut présenter la matrice position par la formule suivante:

$$Q_{i',j'} = \alpha e^{-\gamma(i-i+j-j)} \tag{4.13}$$

La matrice de position doit être modifiée afin d'être concaténée avec le vecteur codant l'histogramme représentant la caractéristique d'apparence. Un vecteur de position est réalisé par l'application d'un balayage en zig-zag sur la matrice de position. Figures 4.20.a, 4.20.b et

4.20.c montrent les vecteurs de position correspondant aux matrices de position des personnes figurants dans 4.19.a.2, 4.19.b.2 et 4.19.c.2, respectivement.

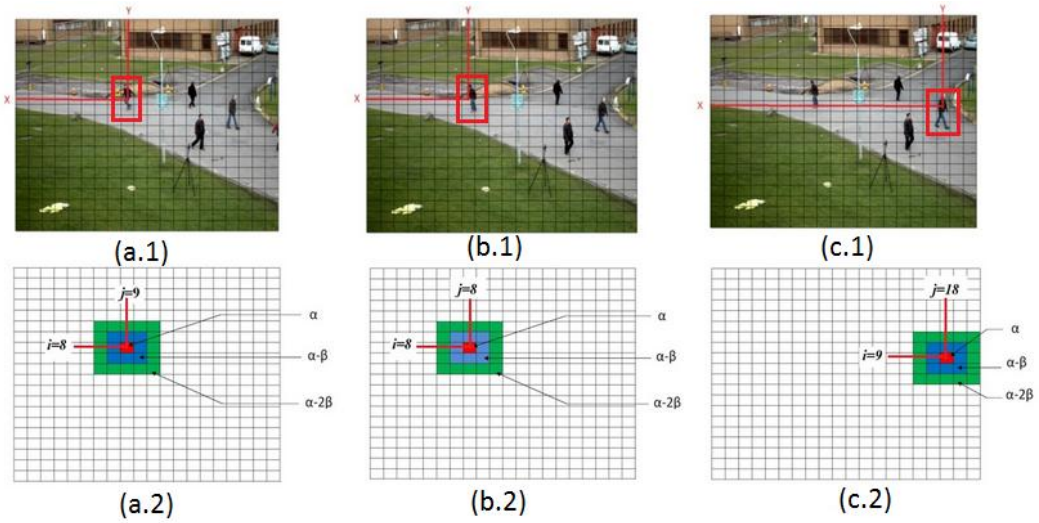


Figure.4.19 Différentes matrices position Q.

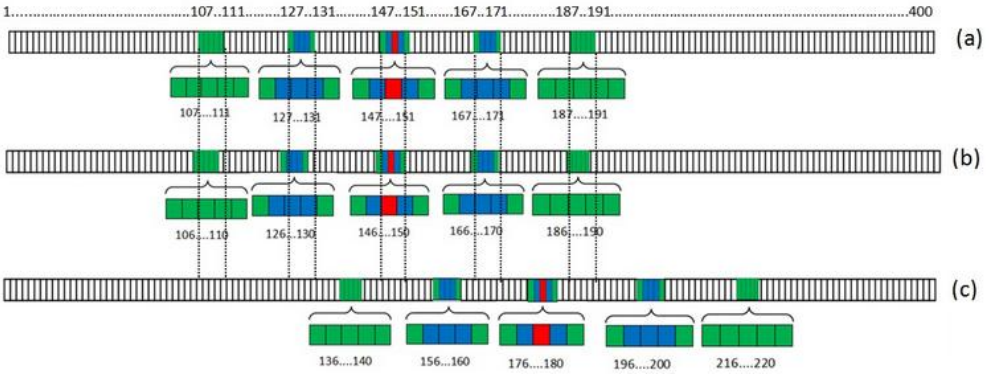


Figure.4.20. Les vecteurs de positions

Nous appelons $Vposition \in \mathbb{R}^{m \times n}$ le vecteur obtenu après un balayage en zig-zag de la matrice de position Q. $E_t = \{1, \dots, P_t\}$ et $E_{t+1} = \{1, \dots, P_{t+1}\}$ représentent l'ensemble des indices des objets détectés à t et t + 1, et $Vposition^l$ le vecteur de position d'objet l. Si $\langle Vposition^{\hat{\imath}}, Vposition^{\hat{u}} \rangle \gg 0$, où $\hat{\imath} \in E_t$ et $\hat{u} \in E_{t+1}$, alors les deux objets se trouvent dans des positions proches de l'autre. Si $\langle Vposition^{\hat{\imath}}, Vposition^{\hat{u}} \rangle \cong 0$, alors la distance entre les positions des deux objets est grande. Par exemple, nous supposons que $Vposition^1, Vposition^2, Vposition^3$ sont les vecteurs de position indiqué par les figures.4.20.a, 4.20.b et 4.20.c, respectivement. On remarque que l'emplacement des zones colorées du vecteur $Vposition^1$ correspond à celui du vecteur $Vposition^2$, et ne coïncide pas avec les cellules colorées de $Vposition^3$. Ainsi, le produit intérieur $\langle Vposition^1, Vposition^2 \rangle \gg \langle Vposition^1, Vposition^3 \rangle$.

IV.3.3 Le descripteur de l'objet détecté

Une fois l'objet détecté, il est représenté par un vecteur codant les informations nécessaires à sa reconnaissance. Ces informations se divisent en deux parties: les caractéristiques de l'apparence et les caractéristiques de la position.

Pour les caractéristiques de l'apparence, un histogramme RGB est construit après avoir divisé le corps en deux parties (le haut et le bas). Cette technique est utilisée pour le simple fait que les personnes généralement mettent des vêtements de différentes couleurs, et la distinction sera meilleure en procédant ainsi. Par exemple, si une personne met une veste de couleur grise et un pantalon de couleur bleue, son histogramme sera similaire à une personne qui met une veste de couleur bleue et un pantalon de couleur grise (le cas inverse), ceci en calculant l'histogramme à partir de la totalité du corps de l'objet; mais en le calculant en divisant le corps en deux parties, la distinction sera meilleure. Ainsi on aura trois vecteurs pour coder les histogrammes de chaque couleurs de chaque partie, pour la totalité du corps on aura six vecteurs : $\{VR^{up} \in \mathbb{R}^{k\times1}, VG^{up} \in \mathbb{R}^{k\times1}, VB^{up} \in \mathbb{R}^{k\times1}, VR_{low} \in \mathbb{R}^{k\times1}, VG_{low} \in \mathbb{R}^{k\times1}, VB_{low} \in \mathbb{R}^{k\times1}\}$ où k désigne le niveau de couleur. On note par VH la concaténation de ces six vecteur (vecteur de l'apparence).

Un descripteur y d'un objet détecté est la concaténation de VH et du vecteur position $Vposition$. Ainsi $y = VR^{up}, VG^{up}, VB^{up}, VR_{low}, VG_{low}, VB_{low}, et\ Vposition$. La figure 4.21 est une représentation graphique du descripteur y.

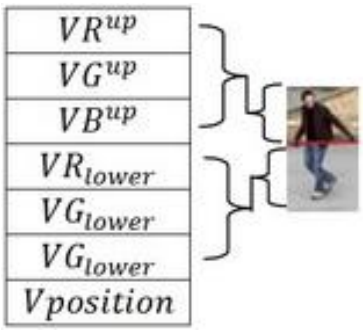


Figure 4.21. Le descripteur d'un objet

IV.3.4 Résumé de l'étape de la détection et de la représentation

Dans ce travail, on a constaté que la soustraction de fond peut être une méthode favorable à la détection du mouvement. Cependant des opérations morphologiques ont été utilisées comme

moyen de filtrage pour améliorer les résultats de cette technique. Une détection de contour est appliquée pour délimiter les zones exactes contenant les objets détectés afin que l'histogramme puisse être plus exact en éliminant les informations appartenant au fond. Chaque objet est représenté par deux histogrammes (haut et bas) ainsi que par un vecteur codant sa position dans la scène. Cette technique est nouvelle et a permis une amélioration des résultats [Elbahri 15]. La figure 4.22 illustre l'étape de détection et de représentation.

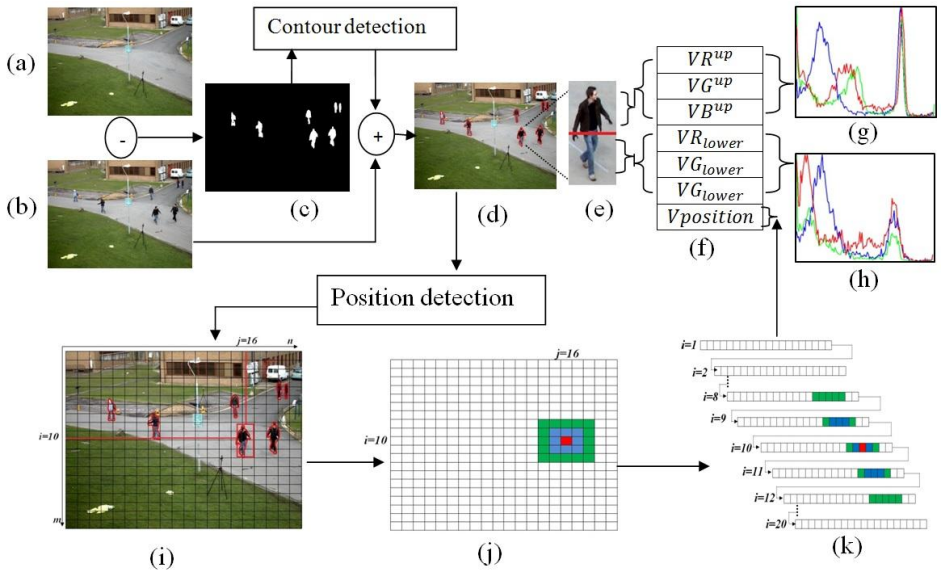


Figure. 4.22. Schéma global pour la détection et la représentation des objets . (a) Le fond référence. (b) scène Observée. (c) Résultat de la soustraction de fond. (d) Silhouettes des objets mobiles. (e) Agrandissement sur un objet détecté. (f) Descripteur d'un objet. (g) et (h) Histogrammes couleur du haut et du bas du corps de l'objet. (i) centre de l'objet détecté au bloque (10,16). (j) Valeurs attribuées au bloques relativement au centre (k) Vecteur contenant le descripteur d'un objet.

Dans certains cas de chevauchement une opération est appliquée pour définir les objets qui chevauchent. Les objets se chevauchant peuvent être considérés comme un; ce cas est détecté lorsque la largeur de la zone de délimitation contenant l'objet dépasse 55% de sa hauteur. Dans le cas où le chevauchement est latéral, nous utilisons le masque pour diviser la zone en deux parties, contenant chacune la forme d'un objet. La répartition des pixels blancs dans la partie supérieure du masque, permet de déterminer le nombre d'objets se chevauchant (Figure 4.23). Le Seuil Δ est réglé afin d'obtenir le nombre de têtes de personnes compris dans la boîte de sélection. Une fois les objets détectés correctement, le programme proposé calcule leur apparence à nouveau.

Figure 4.23 Des objets en chevauchement

IV.4 La reconnaissance et le suivi des objets mobiles

Dans cette partie, nous présenterons les techniques utilisées pour la reconnaissance et le suivi des objets dans une séquence d'image. Nous avons testé en premier lieu des méthodes génératives basées sur l'apparence. Deux approches sont proposées: la première basée sur la mesure de similarité des couleurs, et la deuxième sur la correspondance des points d'intérêt.
On a constaté que ces méthodes donnent de bon résultats lorsqu'il y a peu d'objets et que l'acquisition est de près. Cependant, nous voulons faire un suivi multi-objet et une estimation du vecteur mouvement de tout ce qui bouge dans la scène. Ces deux techniques ne sont pas très robustes dans le cas où la scène est encombrée et surtout lorsque les objets quittent la scène et réapparaissent plus tard. Pour construire un système assez fiable dans ce genre de situation, nous avons utilisé un système de suivi et de reconnaissance basé sur un aspect discriminatif au lieu d'un aspect génératif.

IV.4. 1 Reconnaissance d'objet par les caractéristiques SURF

Grace à sa robustesse contre les variations de l'échelle, de la luminosité, aux mouvements de caméra et aux rotations le SURF[Bay 08] est un moyen très efficace pour extraire les points d'intérêt d'une image (*caractéristiques robustes accélérées)*. Ces caractéristiques sont utilisées pour le recalage des images, la création de mosaïque d'images et pour la reconnaissance d'objets dans une image. Nous avons appliqué cette technique dans deux différents scénarios, le premier comprenant des scènes encombrées de différents personnes mobiles, et le deuxième avec un simple objet mobile. Le résultat de cette technique n'est pas le même pour les deux cas. Nous présentons par la figure 4.24 un exemple de système de reconnaissance basée sur les caractéristique SURF. On constate clairement que le système n'a pas reconnu l'objet cible dans la scène et l'a considéré comme un autre ayant un aspect similaire. L'objet encadré par un rectangle rouge a le plus de caractéristiques similaires à celles de l'objet cible, de ce fait il est élu comme l'objet détecté le représentant dans cette scène acquise à un moment donné.

Pour d'autres genres de scènes moins compliquées (peu d'objet mobile), l'utilisant de ces caractéristiques pour la détection du mouvement ainsi que pour la reconnaissance des objets peut être plus favorable. D'après les exemples montrés dans la section IV.2.2, ces caractéristiques peuvent déterminer les zones contenant un mouvement. dans l'exemple suivant (figure 4.25). Nous avons testé la détection et la reconnaissance d'un objet cible (une boite). Ce processus a réussi en premier lieu la détection. et en suite, pour la reconnaissance il aurait fallu juste comparer les caractéristiques de l'objet cible avec les caractéristiques de l'image acquise.

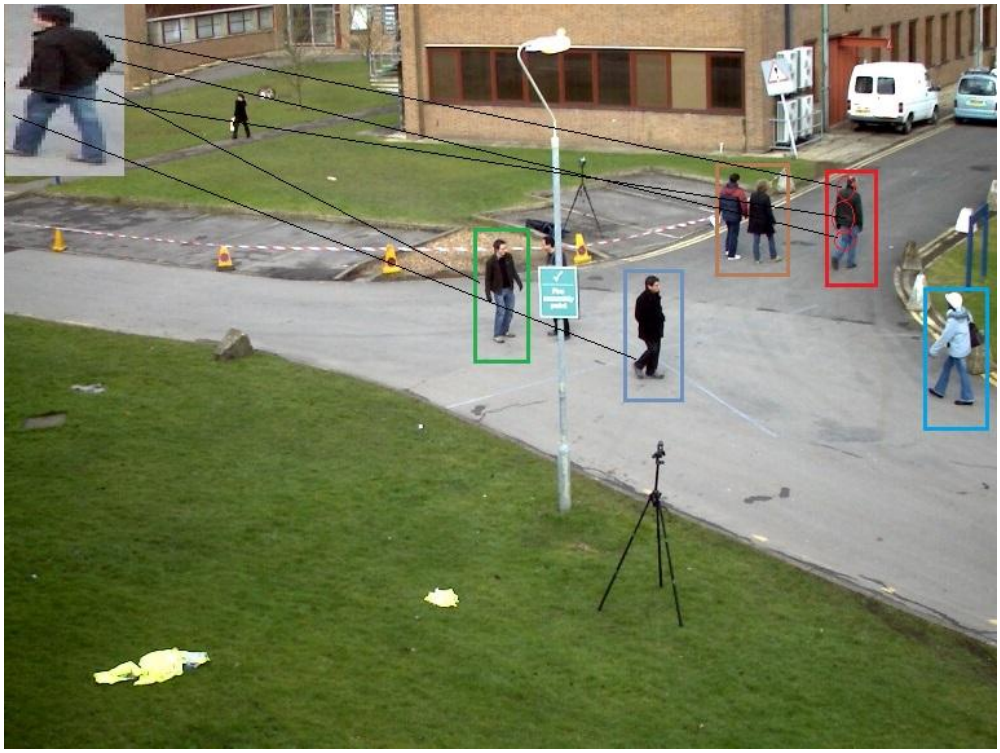


Figure 4.24 Reconnaissance d'objet par les caractéristiques SURF

Sur la figure 4.25.a, nous pouvons voir clairement des lignes colorées qui représentent une correspondance entre les caractéristiques de l'objet (la boite) et une partie de l'image contenant ce même objet. Sachant qu'il n' y a pas uniquement que la boite qui a bougé, mais aussi le fauteuil et le bras. En effet la zone de la scène observée contenant le plus de caractéristiques similaires a celles de l'objet cible, est déterminée comme la zone contenant le même objet cible (voir figure 4.25.b).

IV.4. 2 Reconnaissance d'objets par la mesure de similarité des histogrammes

Dans cette partie, on procède à la reconnaissance d'objets par la mesure de similarité. Pour reconnaitre un objet nommé O_i^t appartenant à l'image I_t dans une autre image I_{t+1} , il faut chercher calculer la corrélation entre l'objet O_i^{t+1} et tous les objets détectés dans I_{t+1}.
Ainsi la reconnaissance peut être formulée comme suit: On note par N_t l'ensemble des objets détectés dans l'image I_t et par N_{t+1} l'ensemble des objets détectés dans l'image I_{t+1}. $i \in N_t$ et les $j \in N_{t+1}$ représentent les indices des objets dans I_t et I_{t+1} respectivement. On aura alors:

$$i = \text{argmax}_j [\![H_i, H_j]\!] \tag{4.14}$$

Où H_i, H_j représentent les histogrammes des objets O_i^t et O_j^{t+1}.

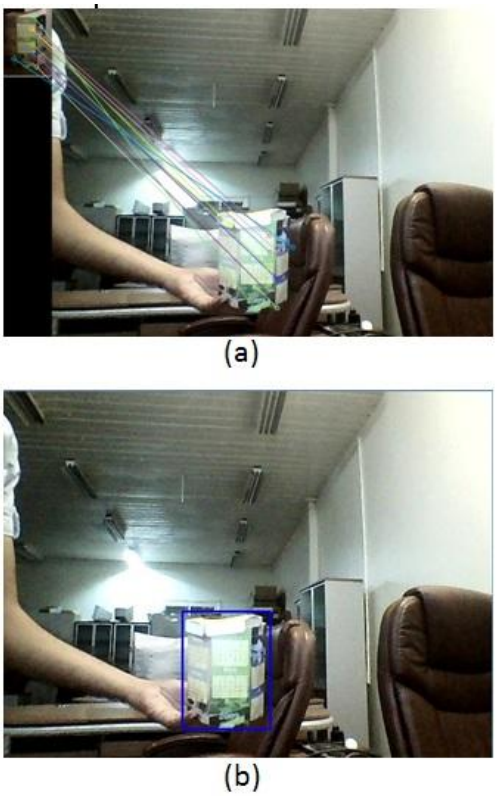


Figure 4.25 Reconnaissance d'objet avec les points SURF

Cependant nous démontrons par un contre exemple tiré par nos expérimentations que cette technique peut faillir dans certains cas. Dans le tableau suivant, nous avons calculé la corrélation entre tous les objets détectés dans deux images. On remarque que la reconnaissance est exacte dans trois cas et la valeur maximale de la corrélation est exacte (représenté en vert), sauf pour un cas ou un autre objet est plus corrélé avec l'objet cible (marqué en rouge).

Tableau 2. Résultat de la corrélation

				
	0,66	0,45	0,23	0,77
	0,57	0,81	0,26	0,45
	0,14	0,21	0,69	0,11
	0,55	0,48	0,30	0,89

De ce fait on peut conclure que la mesure de similitude ne peut être considérée comme une technique très fiable dans ce genre de situation où la luminosité change considérablement.

IV.4. 3 Reconnaissance d'objets en utilisant les représentations parcimonieuses

Dans les sections précédentes (IV.4.1 et IV.4.2) nous avons abordé le problème de reconnaissance avec des caractéristiques accélérées ,robustes et de la mesure de similarité entre les histogrammes. Cependant ces deux techniques ne peuvent pas aboutir à de bon résultats. Nous pouvons formaliser ce problème comme un système de classification discriminatif au lieu de procéder à un système de reconnaissance génératif basé sur l'apparence.

Nous utiliserons pour la reconnaissance un algorithme de représentation parcimonieuse pour la classification et la reconnaissance des objets.

IV.4.3.1 Formalisation du problème de reconnaissance

La reconnaissance peut être formalisée comme un processus de classification grâce à un modèle de régression linéaire. Ce problème peut être résolu par la solution de la minimisation ℓ^1 basée sur une représentation parcimonieuse

$$y = A\beta + \varepsilon \tag{4.15}$$

Résoudre le système linéaire (4.15) peut être la clé pour le problème de la reconnaissance; où $y \in \mathbb{R}^{M\times 1}$ est le descripteur de l'objet mobile et ε est l'erreur tolérée. Ce vecteur contient une information représentative à la description de l'objet défini ci-dessus. Tous les descripteurs d'objets détectés dans les scènes précédentes sont rassemblés dans un dictionnaire $A \in \mathbb{R}^{M\times N}$ (Figure 4.26). Les coefficients de $\beta \in \mathbb{R}^{\mathrm{N}}$sont nécessaires pour déterminer la classe de cet objet.

Figure 4.26 Dictionnaire contenant les descripteurs des objets

IV.4.3.2 La classification basée sur les représentations parcimonieuses (SRC)

L'objectif principal de ce travail est la représentation discriminative des objets détectés et de les indexer avec la même étiquette où qu'ils se trouvent dans la scène, même quand ils la quittent et réapparaissent après dans les futures images. Le système de classification utilisé est basé sur la représentation parcimonieuse et peut être exprimé comme suit:

Le dictionnaire A, contient N descripteurs d'objets recueillis depuis les images précédentes (Ancienne détection). On désigne par no le nombre d'objets étiquetés, nous regroupons ses descripteurs n_i détectés dans les images précédentes pour créer la classe d'objet $A_{ci} = [v_{i1}, v_{i2}, v_{i3}, \dots, v_{in_i}] \in \mathbb{R}^{M\times n_i}$ où $\in \mathbb{R}^{M\times 1}$, $0 < i \leq no$ et $\sum_{i=1}^{no} n_i = N$. En procédant de cette façon, le dictionnaire peut être considéré comme la concaténation de no de classes correspondant no objets marqués:$A = [A_{c1}, A_{c2}, A_{c3}, \dots, A_{c\,no}]$.

Le descripteur y de l'objet de test détecté à partir de la scène courante, peut être approximé par la combinaison linéaire des objets connus que dans la formule (4.15), où $\beta = (\beta_1, \beta_2, \dots\dots, \beta_N) \in \mathbb{R}^{\mathrm{N}}$ est le vecteur de parcimonie. Il est dit k-parcimonieux si $|supp(\beta)| \leq k$, où $supp(\beta) = \{j : \beta_j \neq 0\}$.

La figure 4.27.a, montre un signal d'entrée y à être linéairement approximé. Les figures 4.27.b, 4.27.c, 4.27.d et 4.27.e représentent des classes d'objets étiquetés. Ces classes constituent le dictionnaire A. Ils contiennent des descripteurs des objets détectés sur les images précédentes. Comme on peut le voir dans figure.4.27.b, la classe contient trois vecteurs représentant le même objet à différentes étapes de détection.

La solution parcimonieuse peut être résolue comme un problème de régression linéaire, et peut être formulée par la ℓ^1minimization provenant de la solution parcimonieuse de la formule (4.15):

$$(\ell^1): \hat{\beta} = argmin\{\|y - A\beta\|_2 + \tau\|\beta\|_1\} \tag{4.16}$$

où τ est un paramètre de coût. Pour classer chaque nouvel objet de test représenté par y, nous calculons sa représentation parcimonieuse en minimisant le nombre de coefficients non nuls de vecteur de parcimonie β et la valeur du résidu.

$$\hat{\beta} = argmin\|\beta\|_1 \text{ concernant } \|y - A\beta\|_2 \leq \varepsilon \tag{4.17}$$

Maintenant, pour associer l'objet représenté par y à la ième classe de l'ensemble du dictionnaire d'apprentissage, y peut être approximé par

$$\hat{y}_i = A\beta_i \tag{4.18}$$

où $\beta_i = [0,..,0,\beta_{i1},\beta_{i2},..,\beta_{in_i},0,..,0]$ est un vecteur obtenu après annulation de tous les éléments hors que ceux du *i*ème objet.

La représentation linéaire de y peut être réécrite sous forme:

$$\hat{y}_i = v_{i1}\beta_{i1} + v_{i2}\beta_{i2} + \cdots + v_{in_i}\beta_{in_i} \tag{4.19}$$

Par l'utilisation de cette approximation, tout objet détecté peut être affecté à la *i*ème classe qui satisfait:

$$class(y) = argmin_i\|y - \hat{y}_i)\|_2 = argmin_i\|y - A\beta_i)\|_2 \tag{4.20}$$

où $0 < i \leq no$.

Lorsque les coefficients non nuls sont dispersés sur toutes les classes, et:

$$\max\left(\sum_{l=1}^{n_i}\beta_{il}\right) < \theta\sum_{j=1}^{N}\beta_j \tag{4.21}$$

où $0 < \theta < 1$ est un seuil, alors on peut conclure que l'objet apparait pour la première fois.

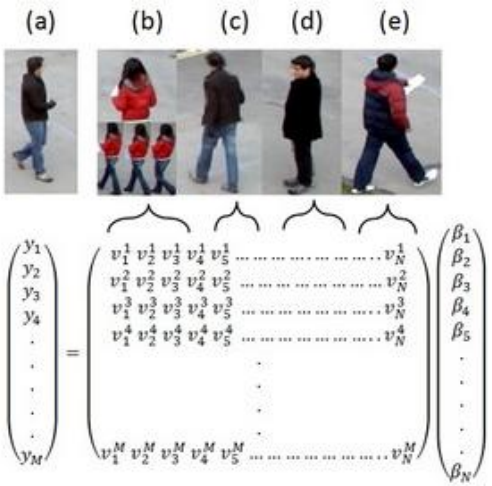


Figure 4.27 Représentation graphique du système linéaire y = Aβ

La figure 4.28 représente un exemple de la solution proposée. Le dictionnaire utilisé pour la reconnaissance contient 30 objets marqués et regroupés en 6 classes (Figure. 4.28.a). Après l'obtention de la solution parcimonieuse de deux objets détectés, les coefficients des vecteurs β correspondants sont représentés sur les figures 4.28.b et 4.28.c. Le premier objet est déjà apparu et existe dans le dictionnaire. Ainsi, ses coefficients de parcimonie sont regroupés dans la deuxième classe (voir la deuxième classe de la figure. 4.28.a). Cependant, l'objet dans la figure. 4.28.c est nouveau, c-à-d qu'il apparait pour la 1ère fois et donc n'existe pas dans le dictionnaire; en conséquence, les coefficients de son vecteur de parcimonie sont dispersés sur de nombreuses classes.

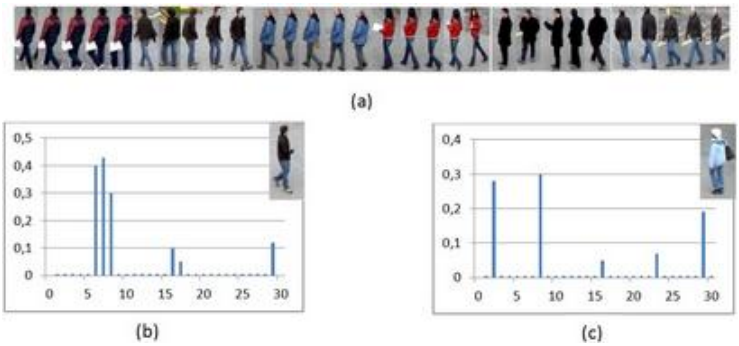


Figure 4.28 Approximation linéaire de deux objets

IV.4.3.3 L'algorithme Orthogonal Matching Pursuit (OMP)

Comme expliqué précédemment, le problème de la reconnaissance consiste à trouver les valeurs des coefficients parcimonieux. Pour obtenir ces valeurs, SRC peut fonctionner comme

un algorithme glouton itératif qui sélectionne à chaque étape, la colonne du dictionnaire A qui génère le plus grand produit intérieur avec les résidus actuels concernant la formule (4.19). Ceci est réalisé par l'algorithme d'OMP décrit ci-dessous.

Dans le domaine des SRC, plusieurs algorithmes sont proposés [Efron 03, Feng 14]. Notre mise en œuvre se base sur l'OMP utilisé pour trouver les coefficients de parcimonie. Cet algorithme a été choisi pour sa rapidité de convergence du fait de la projection orthogonale [Zhang 14, Tony 11]. Dans [Lu 13], Weizhi Lu et al. ont montré que, même lorsque le dictionnaire a des échantillons faux ou bruités, les coefficients de parcimonie peuvent être correctement regroupés en utilisant l'algorithme OMP.

On représente encore une fois l'algorithme OMP cette forme :

Comme entrée, On aura le signal $y \in \mathbb{R}^{M\times 1}$ et la matrice $A \in \mathbb{R}^{M\times N}$ Contenant N colonne. $A(c)$est une sous-matrice de A où $c \subseteq \{1,2,\dots,N\}$ représente un sous-ensemble. on note par t le nombre des itérations et par c_t le sous-ensemble correspondant à la t^{th} iteration. une fois le résidu r_t ou le nombre autorisé d'itération atteint un certain seuil, alors le vecteur $\hat{\beta} \in \mathbb{R}^{t\times 1}$ contient la solution parcimonieuse, où t représente la dernière itération.

Algorithme. 1: Orthogonal Matching Pursuit algorithm

Entrée: y et A.
Sortie: $\hat{\beta}$.
Initialisation: r_0=y, A(c_0)={}.

Pour t = 1; t = t + 1 ***faire***

Etape 1. Trouver la colonne A_i vérifie le problème de maximisation
$argmax_i\langle A_i, r_{t-1}\rangle$ où $i \in \{1,2,\dots,N\} \setminus c_{t-1}$.
Mettre à jour $c_t = c_{t-1} \cup \{i\}$.

Etape 2. Calculer $\hat{\beta} = \left(A(c_t)^T A(c_t)\right)^{-1} A(c_t)^T$
et $P_t = A(c_t)\hat{\beta}$, où P_t désigne la projection sur l'espace linaire décrit par les éléments de $A(c_t)$.

Etape 3. Mettre à jour $r_t = (I - P_t)y$

Etape 4. Si $\|r_t\|_2 < \epsilon$ or t atteint un certain nombre d'itération alors on arrête l'algorithme

Fin .

Une fois que l'objet est classifié, il sera stocké temporairement jusqu'à la fin de la procédure de tous les objets. Après cela, il est ajouté au dictionnaire, plus précisément dans la classe appropriée. Tous les descripteurs appartenant à la même classe seront également modifiés en

mettant à jour la partie de position pour devenir la même que la partie "position" du nouveau descripteur ajouté à leur classe.

IV.4.4 Résultats expérimentaux

Nous présentons dans cette partie, les résultats majeurs obtenus grâce à l'utilisation des représentations parcimonieuses et notre contribution pour la description des objets a travers leur apparence et leur position. Nous avons choisi une vidéo contenant plusieurs personnes qui bougent et qui sont habillées avec des couleurs presque similaires.

IV.4.4.1 description de la vidéo référence

Nous avons testé notre algorithme sur la vidéo PETS '09. Cette vidéo est filmée pour être une référence dans le suivi de d'objet et elle est utilisée dans de nombreuses approches. Cette vidéo est encombrée, ainsi la reconnaissance devient un grand défi parce que les gens portent des vêtements avec des couleurs similaires. Cette vidéo est constituée de 800 images, 10 personnes apparaissent dans cette vidéo et doivent être indexées à leur première apparition dans la scène reconnues quand elles quittent la scène et réapparaissent à nouveau.

IV.4.4.2 Discussion et comparaison des résultats visuels

Nous présentons dans cette partie une discussion sur des résultats visuels de notre approche et d'autre méthode de l'état de l'art: MTUSR [Lu 13], EPFL [Ben 11] et ETHZ [Breitenstein 11]. Toutes les personnes sont correctement initialisées et reconnus [Vidéo]. elles sont indexés à partir de 0 selon l'ordre de leur apparition dans la vidéo. L'utilisation du vecteur de position présentée dans la section 4.3.2, contribue quand les couleurs sont similaires. Dans Figure.4.29.a le descripteur utilisé pour cet exemple est construit uniquement avec des histogrammes de couleurs. Lorsque certains objets sont éclipsés (personne numéro 2 sur la partie gauche de Fig.4.29.a est masqué sur l'image à droite), il ya une certaine confusion entre cette personne et un autre qui apparaît pour la première fois sur l'image droite de la figure.4.29. La figure.4.29.b est un exemple montrant l'efficacité de l'ajout du vecteur de position dans notre descripteur. En outre, le vecteur de position rend le système de suivi plus robuste dans le cas de l'occlusion partielle (voir figure.4.30). Dans la figure.4.31 la méthode proposée traite avec succès et a permis la détection même en cas de chevauchement et reconnait autant que possible les objets en mouvement. Dans frame_0160 de la figure.31,

deux objets ne sont pas détectés par MTUSR à cause du chevauchement, mais ils sont bien détectés par l'ETHZ, EPFL et la méthode proposée. Dans frame_0031 de la même figure, deux personnes sont considérées comme un seul par MTUSR, mais sont bien séparées par l'ETHZ, EPFL et notre méthode. La raison en est que, dans l'approche de MTUSR, lorsque la taille du cadre de sélection contenant l'objet en mouvement est trop grande, l'algorithme évite simplement ce cas de détection où deux objets sont considérés comme un, s'il se chevauchent un peu; tandis que nous avons résolu ce genre de problème, comme expliqué dans la section IV.3.4. La similitude des couleurs de vêtements provoque une confusion entre les objets qui se croisent. Certaines méthodes ne sont pas très robustes pour gérer ces situations. Comme on peut le voir sur la figure 4.32, le recouvrement inverse des indices des objets 6 et 8 et entre frame_0675 et frame_0667 en utilisant la méthode EPFL basée sur les trajectoires des objets en mouvement; il est mentionné dans EPFL que leur approche n'est pas applicable lorsque les mouvements des objets sont très irréguliers. L'ETHZ, MTUHZ et l'approche proposée basée sur un système d'apprentissage, semblent robustes dans ces cas.

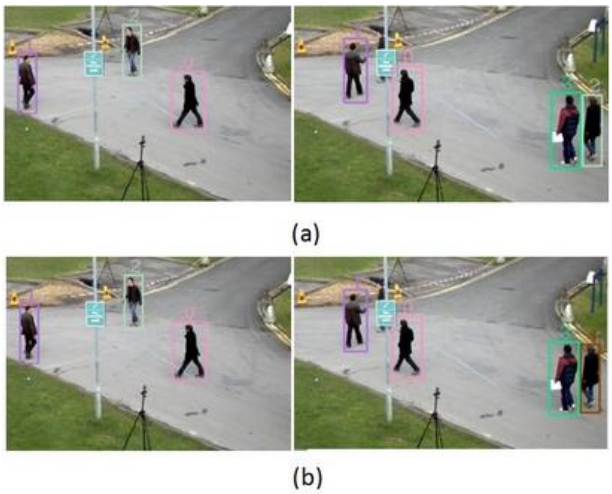


Figure. 4.29 Reconnaissance d'objets avec et sans la position

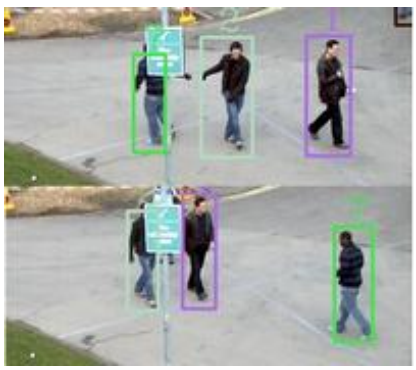


Figure 4.30 Problème d'occlusion

Lorsque les objets quittent la scène pour réapparaître de nouveau dans les futures scènes, le système doit les reconnaitre. Dans la figure.4.33 notre méthode, maintient l'indice de la

personne 5 dans frame_0063 et frame_0471. L'approche de MTUSR fonctionne bien aussi. Pour MTUHZ et notre méthode qui utilisent les caractéristiques d'apparence et la position; pour l'ETHZ qui utilise un cadre de filtrage de particule et pour la méthode de l'EPFL, nous pouvons voir que le processus de reconnaissance est faux, et la même personne est marquée par des différentes couleurs dans les scènes. Dans la figure.4.34, les personnes indexées par 3 et 4 dans frame_0031 sont bien récupérées dans frame_0535 et frame_0626 par la méthode proposée. MTUSR ne reconnait pas les personnes indexées par 4 et 5 dans frame_0535 et elles sont considérées comme une nouvelle personne indexée 10. Dans frame_0626, seule la personne indexée 4 est récupérée par MTUSR. ETHZ ne détecte pas la personne en frame_0535 et permute l'identité de deux personnes dans frame_0626. EPFL ne récupère pas les personnes correctement entre frames_0031 et les autres cadres. La raison qui rend notre méthode plus efficace est la façon de représenter les objets comme décrit dans la section IV.4.3.

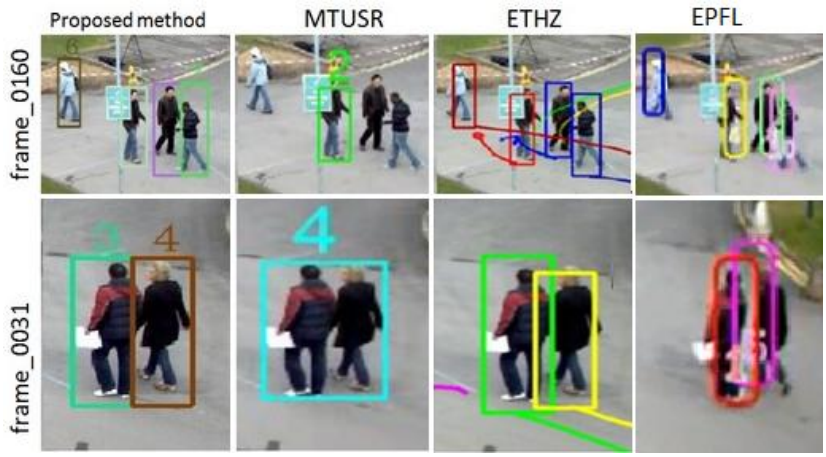


Figure.4.31 Comparaison visuelle du problème de chevauchement

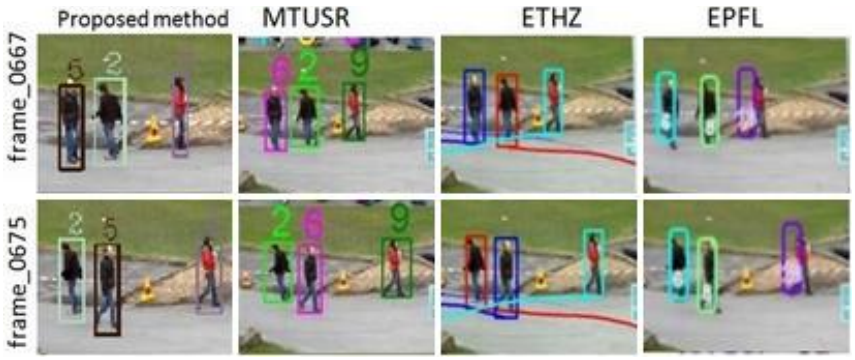


Figure.4.32 Illustration du problème de l'identité double

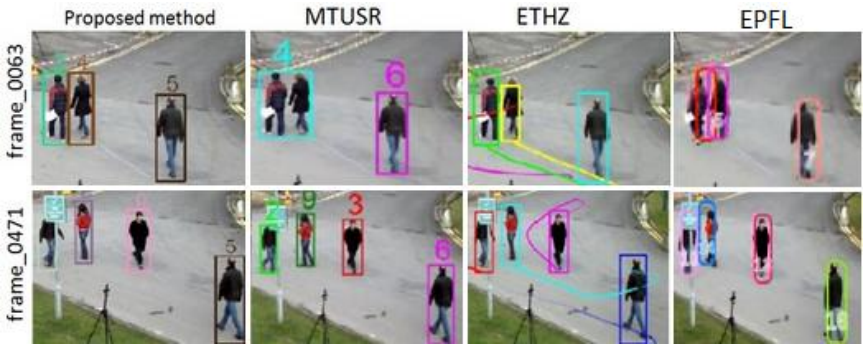


Figure.4.33 Illustration de la reconnaissance des objets

Figure 4.34 Illustration du problème d'initialisation

IV.4.4.3 Evaluation des résultats

Le tableau 3 résume le nombre des entrées de personnes et de la réapparition de la scène. L'approche proposée semble robuste pour une bonne reconnaissance. Selon le tableau 2, MTUSR, EPFZ et l'EPFL ont respectivement 2, 6 et 7 erreurs d'initialisation et de récupération. En plus des comparaisons visuelles, nous utilisons certaines mesures de performance [Bernardin 08]: la précision du multiple suivi d'objets (MOTP) l'exactitude du multiple suivi d'objets (MOTA). MOTP vise à évaluer les trajectoires positifs des cibles, alors que MOTA, composé du taux des résultats faux négatifs, et du taux des résultats faux positifs et le nombre de permutation d'identité, mesure la précision. Nous avons mesuré à la fois le

MOTP et le MOTA sur le résultat de notre approche appliquée sur la vidéo référence . Nous avons obtenu 70% de précision (MOTP) et 85% d'exactitude (MOTA). L'utilisation de la soustraction de fond en tant que technique de détection a ses limites en particulier lorsque l'objet est caché par un autre ou lorsque deux objets sont très proches l'un de l'autre. Ces cas de mauvaise détection ont comme effet de réduire le MOTP de notre approche à 70%. La métrique MOTA de notre approche est également affectée par ces limitations lors de l'étape de détection, mais notre système de suivi basé sur les représentations parcimonieuses pour la reconnaissance d'objet fonctionne très bien en comparaison avec d'autres approches. le tableau 4 montre les métriques MOTP et MOTA de notre approche et les compare avec d'autres travaux utilisant la même vidéo de référence. En se basant sur ce tableau, on peut conclure que l'approche proposée donne des résultats satisfaisants malgré la limitation de soustraction de fond.

Tableau. 3: Entrée des objets et réapparitions dans la scène, la valeur F signifie fausse initialisation. La dernière colonne (NE) contient le nombre d'erreurs de chaque méthode.

Objets	0	1	2	3	4	5	6	7	8	9	
Nombre des entrées	2	4	2	2	2	2	1	1	2	1	
Méthode	Entées détectées										NE
EPFL	1	2	1	1	1	1	1	1	2	1	7
MTUSR	2	4	2	1	1	2	1	1	2	1	2
ETHZ	2	4	2	1	1	1	1	1	F	F	6
Approche proposée	2	4	2	2	2	2	1	1	2	1	0

Tableau. 4: Evaluation des résultats MOTP et MOTA

Methode	**MOTP**	**MOTA**
ETHZ	52%	50%
EPFL	<50%	<50%
Possegger *et al*	64,8%	66%
Berclaz *et al*	50%	58%
Wu *et al*	73,3%	73,2%
Proposée	70%	85%

IV.4 L'estimation de mouvement

L'étape précédente de notre système basée sur les représentions parcimonieuses nous a permis de bien détecter et reconnaitre chaque objet en mouvement. Pour estimer le vecteur

mouvement de chaque objet, nous pouvons maintenant utiliser les coordonnées du centre de cet objet détecté à chaque image pour reconstruire son vecteur mouvement.

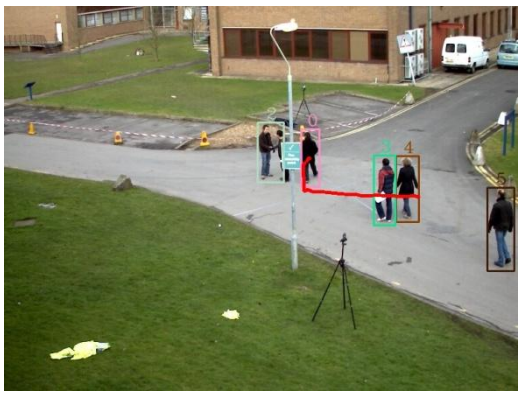
Entre une image I_t et I_{t+1} le vecteur mouvement V^i de l'objet O^i peut être défini comme suit:

$$V^i = \begin{pmatrix} o_x^{i+1} - o_x^i \\ o_y^{i+1} - o_y^i \end{pmatrix} \tag{4.22}$$

Notons que nous nous intéressons à une estimation basée région et non pixèlique (flot optique). Dans les figures suivantes nous pouvons voir clairement le parcours d'un objet au cours de la progression de la vidéo.

(a)

(b)



(c)

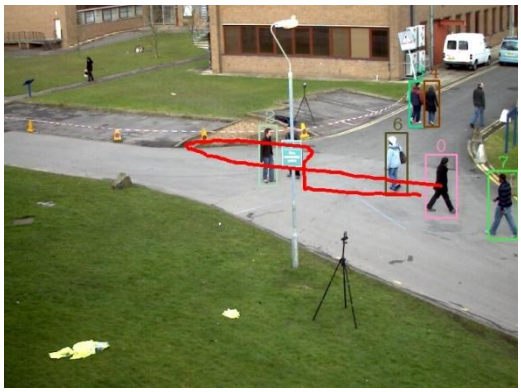


Figure 4.35: Tracé du vecteur mouvement d'un objet. (a) image acquise à un instant t1. (b) image acquise à un instant t2. (c) image acquise à un instant t3

IV.5 Conclusion

Le but de ce travail est la création d'un système de suivi multi-objets capable de détecter le mouvement et surtout de reconnaitre et tracer le parcours de chaque objet. Après avoir testé plusieurs approches génératives qui n'ont pas traité les scènes compliquées, nous nous sommes intéressés aux représentations parcimonieuses pour la création d'un système de classification pour la reconnaissance des objets. L'utilisation de la représentation des objets proposée dans ce travail a permis d'améliorer les résultats. Pour augmenter encore l'exactitude de notre système de suivi, un dictionnaire d'apprentissage en ligne est utilisé. Avec l'algorithme OMP, on peut ainsi générer les coefficients de parcimonie nécessaires à la classification. Une comparaison de notre méthode avec d'autres démontre l'efficacité de notre approche.

Chapitre V

Approche parallèle utilisée

V.I Introduction

Le but de ce travail, est la création d'un système de détection et de suivi d'objet mobile dans une séquence vidéo fiable et exacte. En introduisant les représentations parcimonieuses comme outil de classification et de reconnaissance et la manière proposée pour la représentation des objets, les performances ont augmenté sauf que le temps d'exécution a augmenté aussi.

Dans plusieurs travaux de l'état de l'art les auteurs sacrifies la précision pour accélérer le système ou assure la précision en alourdissant le système et augmentant le temps d'exécution.

Nous proposons dans le présent travail, de garder les performances de notre système de suivi tout en assurant le traitement en temps réel [Elbahri 15,2]. Pour arriver à ces fin, nous avons proposé d'utiliser une implémentation parallèle pour accélérer le traitement. Nous considérons les étapes de détection et de filtrage comme un prétraitement qui ne va pas être parallélisé. Par contre, l'étape de reconnaissance basée sur les représentations parcimonieuses, va être parallélisée. Plus précisément, l'algorithme OMP inclut des opérations matricielles et des produits vectoriels qui alourdissent le système.

Initialement, les GPU ont été utilisées pour l'accélération du graphisme. Ces cartes qui se composent de centaines de cœurs de processeurs, sont maintenant conçus pour être des cartes de processeur qui peuvent exécuter le paradigme parallèle (SIMD). Par conséquent, les GPU sont utilisés dans les ordinateurs personnels, postes de travail et d'autres équipements comme des unités de traitement. En 2007, NVIDIA a proposé le modèle de programmation parallèle de l'architecture de calcul unifiée-périphérique (CUDA) [Lindholm 08]. Sous l'architecture CUDA, la mise en œuvre de programmes parallèles est tout à fait transparente, qui permet à tout programmeur sans une forte connaissance des concepts graphiques à utiliser ces cartes pour une parallélisation SIMD.

Contrairement aux autres plates-formes parallèles, CUDA facilite la programmation et offre une très grande transparence. La fonction qui exécute le calcul parallèle sur l'appareil (GPU) est appelé un noyau. Lorsque le noyau est lancé, de nombreux processus léger (Thread) sont regroupés en blocs et exécutés en parallèle. En outre, une collection de blocs est appelée une grille. Pour lancer le noyau, la dimension de la grille doit être spécifié. CUDA propose au programmeur le choix d'une grille unidimensionnelle ou un bidimensionnelle pour regrouper les threads, et une fois que le noyau est lancé, ses dimensions ne peuvent pas être changées. Le bloc est identifié en utilisant le terme (blockId.x) pour la grille à une dimension,

et par (blockId.x, blockId.y) pour une grille à deux dimensions. Les threads (ThreadID) sont unidimensionnel, bidimensionnel ou tridimensionnel selon la manière avec laquelle les blocs sont: un, deux, ou trois dimensions. La figure 5.1 représente une grille à deux dimensions. Le bloc (1,1) a son blockId.x = 1 et blockId.y = 1. Le thread marqué par (2,1,0) a son threadId.x = 2, threadId.y = 1 et threadId.z = 0.

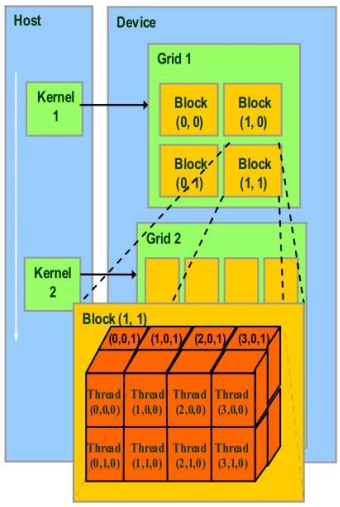


Figure 5.1: Exemple de grille CUDA

Le nombre maximum d'unités d'exécution dans chaque bloc dépend des caractéristiques du matériel. Par exemple, le GPU utilisé dans ce travail est GeForce GT 630M, le maximum de thread par bloc est de 1024. La flexibilité globale de la distribution de ces threads dans les trois dimensions ne peut pas dépasser 1024; donc le produit des indices d'un thread doit être inférieur à 1024. Par exemple, les threads (15,20,3) et (500,2,0) sont admissibles, tandis que le thread (15,20,4) n'est pas admissibles.

V.2 Description de la carte GPU utilisée

La carte utilisée est une NVIDIA GeForce GT 630M intégré dans un ordinateur portable Dell. Elle est dotée d'une mémoire globale de 1Go et de 96 CUDA cores avec une fréquence de 1,25 Ghz. Elle peut supporter une taille maximale de bloc allant jusqu'a1024 x 1024 x 64 et une grille de 65535 x 65535 x 65535. La figure 5.2 est un imprimé écran du résultat de la configuration de CUDA pour cette carte

```
Device 0: "GeForce GT 630M"
  CUDA Driver Version / Runtime Version          5.0 / 4.1
  CUDA Capability Major/Minor version number:    2.1
  Total amount of global memory:                 1024 MBytes (1073610752 bytes)
  ( 2) Multiprocessors x (48) CUDA Cores/MP:     96 CUDA Cores
  GPU Clock Speed:                               1.25 GHz
  Memory Clock rate:                             2000.00 Mhz
  Memory Bus Width:                              64-bit
  L2 Cache Size:                                 131072 bytes
  Max Texture Dimension Size (x,y,z)             1D=(65536), 2D=(65536,65535), 3D=(2048,2048,2048)
  Max Layered Texture Size (dim) x layers        1D=(16384) x 2048, 2D=(16384,16384) x 2048
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per block:           1024
  Maximum sizes of each dimension of a block:    1024 x 1024 x 64
  Maximum sizes of each dimension of a grid:     65535 x 65535 x 65535
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and execution:                 Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Concurrent kernel execution:                   Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support enabled:                No
  Device is using TCC driver mode:               No
  Device supports Unified Addressing (UVA):      No
  Device PCI Bus ID / PCI location ID:           1 / 0
```

Figure 5.2 La configuration de la carte GPU

V.3 L'implémentation parallèle de l'algorithme OMP

De nombreuses implémentations parallèles de l'OMP sont proposées dans la littérature. Chacun d'entre eux se concentre sur la multiplication et à l'inverse des matrices pour réduire le temps du traitement, la première étape de l'OMP qui consiste à calculer la projection sur l'espace linéaire est parallélisable. Selon les résultats obtenus, l'efficacité de ces implémentations dépend du niveau de parcimonie. Pour la compensation des signaux, l'algorithme OMP exige un haut niveau de parcimonie pour la convergence du résidu exprimé par l'équation (4.20). Si le niveau de parcimonie est élevé, la taille des matrices nécessaires pour calculer la projection P_t est trop élevée. Dans ce cas, l'efficacité du calcul sur GPU est importante. Alors que, pour le suivi de l'objet, le niveau élevé de parcimonie n'est pas forcément nécessaire pour classer un objet test désigné comme entrée y. Cela signifie que les tailles des matrices ne sont pas très grande. Toutefois, la taille du dictionnaire peut être très importante car elle augmente à chaque mise à jour. Nous vous proposons ici une parallélisation de la première étape de l'OMP, qui consiste à sélectionner la colonne la plus corrélée avec le dictionnaire de descripteur y de l'objet de test d'entrée. Cette approche est

choisie pour réduire le temps de traitement lorsque la taille du dictionnaire est très grande. En outre, afin de réduire le temps d'accès en mémoire, nous nous proposons de lancer le noyau parallèle pour tous les objets détectés en même temps en parallèle. La parallélisation des calculs d'inverse des matrices n'est lancé que lorsque le niveau k de parcimonie est élevé. Nous démontrons par la suite l'efficacité de cette approche avec des tableaux contenant le temps d'exécution de chaque étape.

Cependant, deux types de programme parallèle vont être discutés: Le premier ressemblant à celui utilisé par [Fang 11, Dai 14]. Le deuxième type que nous proposons est plus adéquat pour un système de suivi qui nécessite un niveau faible de parcimonie mais une taille de dictionnaire importante.

V.3.1 Première approche parallèle

Cette approche consiste à traiter les objets détecté un après l'autre (Figure 5.3). Pour chaque objet détecté on lance les étapes de l'algorithme OMP ainsi chaque étape est parallélisée. La 1ère étape consiste à trouver l'atome du dictionnaire le plus corrélé avec le signal $y \in \mathbb{R}^M$ d'entrée. Ce dictionnaire a une taille $M \times N$. L'étape de sélection se résume en calculant le produit intérieur de chaque colonne du dictionnaire avec le signal y. On aura ainsi $M \times N$ produits et M additions à exécuter (Figure 5.4) pour réaliser cette opération. Pour réaliser cette étape en parallèle, on aura besoin de $M \times N$ threads à lancer. Pour la deuxième étape, elle consiste à faire des calculs matriciels sur une matrice $\grave{A} \in \mathbb{R}^{k \times k}$, où k désigne le degré de parcimonie. Cette approche a de plus grandes performances lorsque le niveau k est élevé et lorsqu'il n'y a qu'un signal d'entrée. Le chargement des données depuis le CPU vers le GPU prend beaucoup de temps et risque de ralentir le programme, ce qui nous ramène à modifier cette approche, pour améliorer le temps d'exécution.

Le temps d'exécution pour la première étape dépend de la taille du dictionnaire. Plus le dictionnaire est grand plus le rapport entre l'exécution parallèle et l'exécution séquentielle sera important. Les résultats sont résumés dans le tableau 5.

Tableau 5. Temps d'exécution de l'étape de la sélection (1ère étape)

Taille du dictionnaire	GPU (ms)	CPU (ms)	Apport (CPU/GPU)
100	0,9	3	3,33
200	0,93	5	5,37
300	0,98	9	9,18
400	1,06	12	11,32
500	1,92	17	8,85
600	1,94	20	10,30
700	2,06	23	11,16
800	2,14	29	13,55
900	2,89	32	11,07
1000	2,92	34	11,64

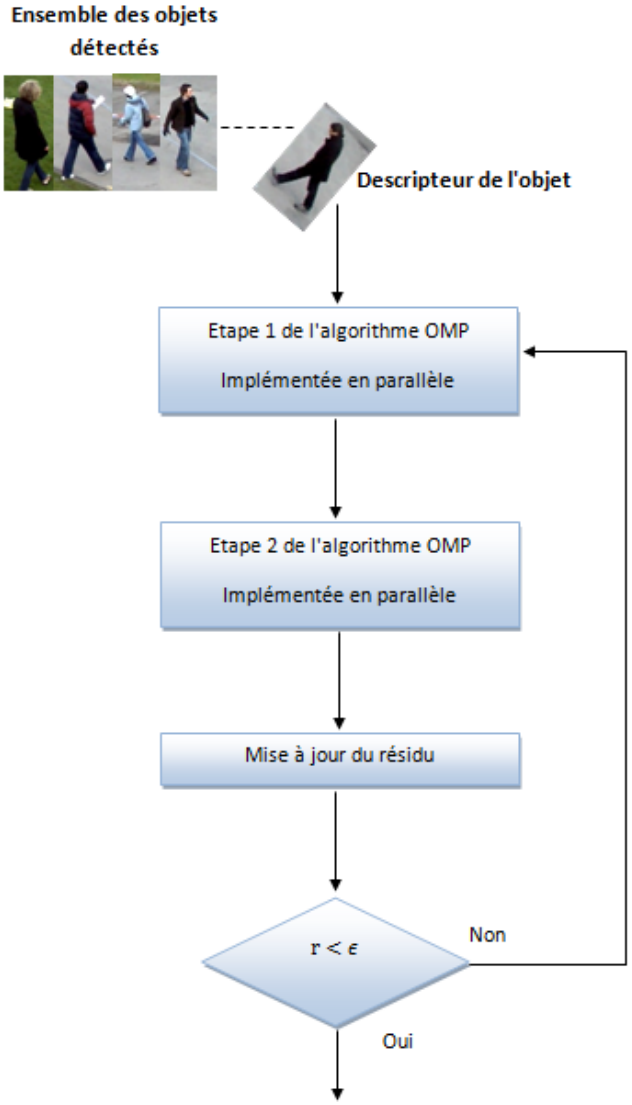


Figure 5.3 Organigramme de la première approche parallèle

```
int tid=threadIdx.x;
int bid=blockIdx.x;
int offset=bid*blockDim.x+tid;
__shared__ float data_f[share_size];
int end=0;
float temp=0.0;
for(int start=0;start<t;start+= share_size)
{
    __syncthreads();
    for(int i=tid;i< share_size && i+start<t;i+=blockDim.x )
    {
        data_f[i]=f[start+i];
    }
      __syncthreads();
    if(offset<t)
     {
        end=start+ share_size >t ? t:start+ share_size;
        for(int i = start; i < end; i++)
        {
              temp += H_inv[i*pitch_H_inv+offset]*data_f[i-start];
        }
         __syncthreads();
        z[offset] = temp;
        }
}
```

Figure 5.4 Exemple de code source

Pour la deuxième étape qui consiste calculer l'inverse de la matrice nécessaire pour l'obtention de la projection orthogonale, le gain dépend du degré de parcimonie k. Contrairement à la compensation des signaux, la classification ne nécessite pas un degré élevé de parcimonie. Donc cette étape peut être exécuté en séquentiel. Le tableau 6 montre les résultats obtenus.

Tableau 5 : Temps d'exécution pour le calcul de l'inverse d'une matrice

Degré de parcimonie	8	128	256	384
Temps CPU(ms)	***12***	5,447	18,880	60,893
Temps GPU(ms)	***23***	150	1,596	7,335

Il est claire d'après le tableau que l'approche est fiable lorsque le niveau de parcimonie dépasse la valeur 100. Dans notre cas il dépasse rarement la valeur 20. On remarque que pour un degré égale à 8, l'implémentation séquentiel est plus rapide que l'implémentation parallèle.

V.3.2 Deuxième approche parallèle

Cette implémentation est différente de la première, elle traite la première étape pour tous les objets en parallèle (voir figure 5.5) contrairement à l'approche précédente qui traite les objets

un après l'autre. Cette technique nous permet de réduire considérablement le temps en réduisant le temps de chargement des données entre le CPU et le GPU.

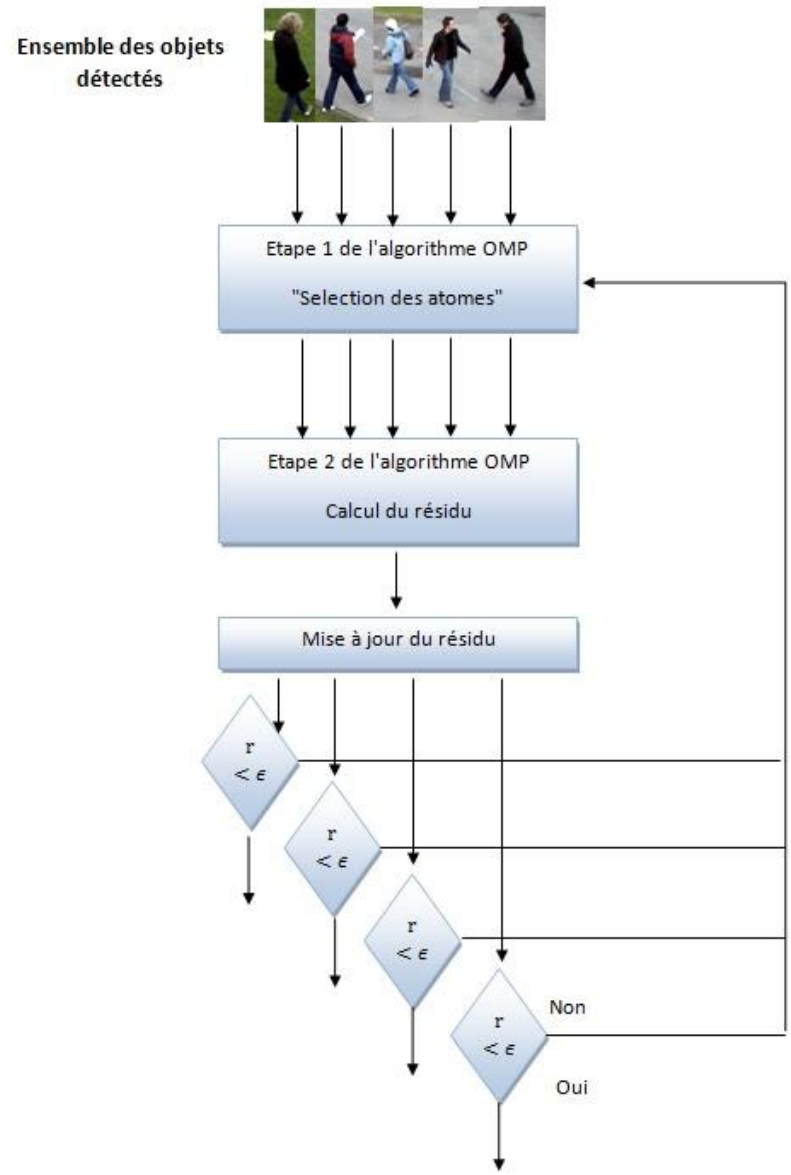


Figure 5.5 Organigramme de la deuxième approche parallèle

L'algorithme OMP parallèle peut être présenté comme suit: Comme entrée, tous les descripteurs des objets y_j où $j \in \{1, 2, \dots p\}$ sont regroupés dans la matrice $y^p \in \mathbb{R}^{M \times p}$ où p est le nombre d'objets détectés. La matrice $A \in \mathbb{R}^{M \times N}$représente le dictionnaire et contient N colonnes. $A(c^j)$est une sous-matrice de A et les indices du sous-ensemble $c^j \subseteq \{1,2, \dots, N\}$. On note par t_j le nombre d'itérations et par $c^j_{t_j}$le sous-ensemble correspondant à la ${t_j}^{ème}$itération. X représente un sous-ensemble des indices des objets.

Algorithme. 2: Orthogonal Matching Pursuit algorithm Parallel

Entrée: y^p et A.

Sortie: $\beta_j^{t_j}$.

Initialisation: $X = \{1, 2, \dots p\}$. $pour\ chauqe\ j \in \{1, 2, \dots p\}$: $r_0^j = y_j$, $A(c_0^j) = \{\}$, $t_j = 1$.

Etape 1. Trouver la colonne A_i qui vérifie le problème de maximisation

$Pour\ chaque\ j \in X,\ c_{t_j}^j = c_{t-1_j}^j \cup\ argmax_i \langle A_i, r_{t_j - 1_j}^j \rangle$ où $i \in \{1,2, \dots, N\} \setminus c_{t-1_j}^j$

Stage 2. $\boldsymbol{for\ each}\ j \in X$ do

Calculer $\beta_j^{t_j} = \left(A\left(c_{t_j}^j\right)^T A\left(c_{t_j}^j\right) \right)^{-1} A\left(c_{t_j}^j\right)^T$ and $P_t^j = A\left(c_{t_j}^j\right) \beta_j^{t_j}$, où P_t^j désigne la projection sur l'espace linaire décrit par les éléments de $A\left(c_{t_j}^j\right)$ correspondant au signal y_j. Mettre à jour $r_{t_j}^j = \left(I - P_{t_j}^j\right) y_j$.

If $\left\| r_{t_j}^j \right\|_2 < \epsilon$ or t_j Atteint le maximum d'iterations, alors $X = X - \{j\}$, sinon, mettre $t_j = t_j + 1$.

$\boldsymbol{Fin\ pour}$

Etape 3. si X={}, Arrêter l' algorithme; sinon, retourner à **étape1**.

Comme sorties, on aura les vecteurs: $\{\beta_1^{t_1} \in \mathbb{R}^{t_1 \times 1}, \beta_2^{t_2} \in \mathbb{R}^{t_2 \times 1}, \dots, \beta_p^{t_p} \in \mathbb{R}^{t_p \times 1}\}$ contenant la solution parcimonieuse. Pour plus d'explication, le sous-ensemble X contient les indices de tous les objets. Lorsque le résidu de l'objet converge en respectant l'équation (4.20), la solution parcimonieuse est satisfaite (4.16). Ensuite, cet objet ne sera plus pris en compte par l'algorithme, et donc il est éliminé de l'ensemble X. La première étape de l'algorithme de OMP est parallélisée comme suit: en utilisant l'architecture CUDA, un noyau parallèle est lancé avec une grille à deux dimension (Figure.5.6.b). Chaque thread dans le bloc dispose également d'indices bidimensionnels (threadId.x, threadId.y). L'indice threadId.x désigne l'indice j de l'objet y_j dans la matrice y^p. Indice threadId.y désigne l'indice i de la colonne $A_i$de la matrice A. Chaque thread (threaded.x, threadId.y), calcule le produit intérieur du $j^{\text{threadId.x}}$descripteur d'objet avec la colonne $A_{\text{threadId.y}}$. Figue. 5.6.a. montre le thread (0, 2) destiné à calculer le produit intérieur de l'objet numéro 0 avec la colonne 3 du dictionnaire A. Le nombre total de threads exécutés est $p \times N$. Comme mentionné dans le début de cette section, le nombre maximal de threads par bloc est limité à 1024 pour la carte utilisée. Nous avons limité dans ce travail le maximum des indices pour chaque thread à (5, 200). Ainsi, si $(p \times N) > (5 \times 200)$, un bloc ne sera pas suffisant pour satisfaire tous les objets et les colonnes. Pour cela, la dimension de la grille dépend du nombre d'objets et de la taille de la

matrice A. Pour avoir suffisamment de threads, le noyau est lancé avec une grille à deux dimensions avec une taille de (gridDim.x) x (gridDim.y). Dans ce cas, gridDim.x = ⌈p / 5⌉ et gridDim.y = ⌈n / 200⌉. Par exemple, si le système détecte 15 personnes et que le dictionnaire contient 1000 colonnes, une grille de (3 × 5) blocs est nécessaire pour exécuter (15 × 1000) threads.

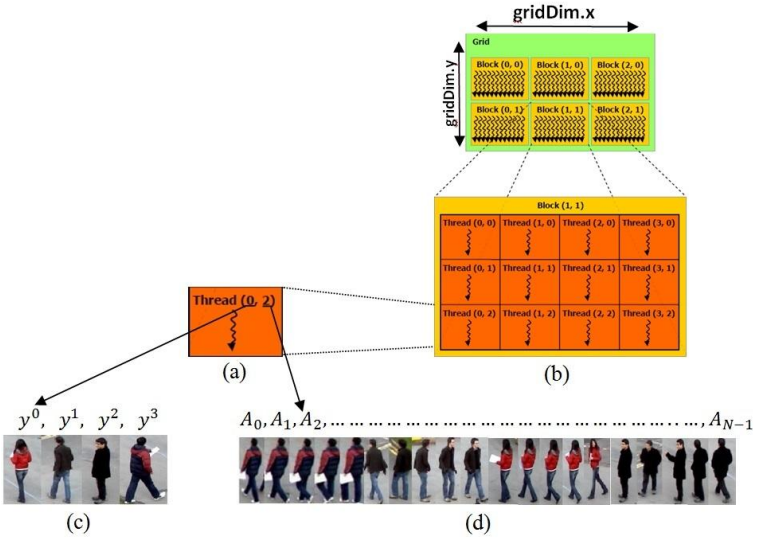


Figure 5.6 Exécution des threads

V.3.3 Temps d'exécution et résultats expérimentaux

On constate clairement que le temps d'exécution de l'approche parallèle est considérablement réduit à $O(\log MN)$. On est arrivé à un système temps-réel permettant de traiter 22 images par seconde, alors que d'autres méthodes exécutent leurs algorithmes plus lentement. D'après la figure 5.7, on remarque que plus le nombre de personne détecté augmente ainsi que la taille du dictionnaire plus le rendorment est plus grand; ceci est dû au nombre d'opérations exécutées en parallèle.

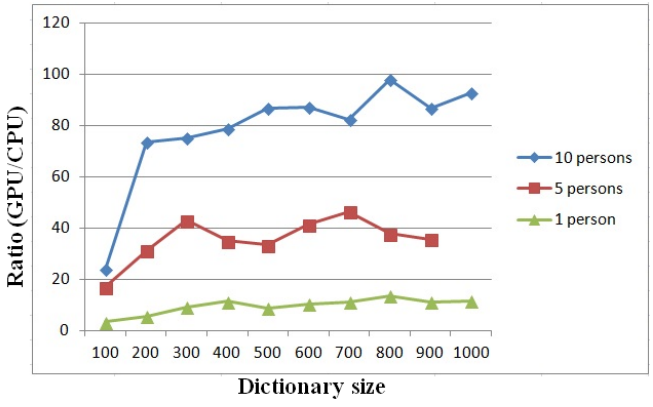


Figure.5.7 L'apport de l'approche parallèle utilisée

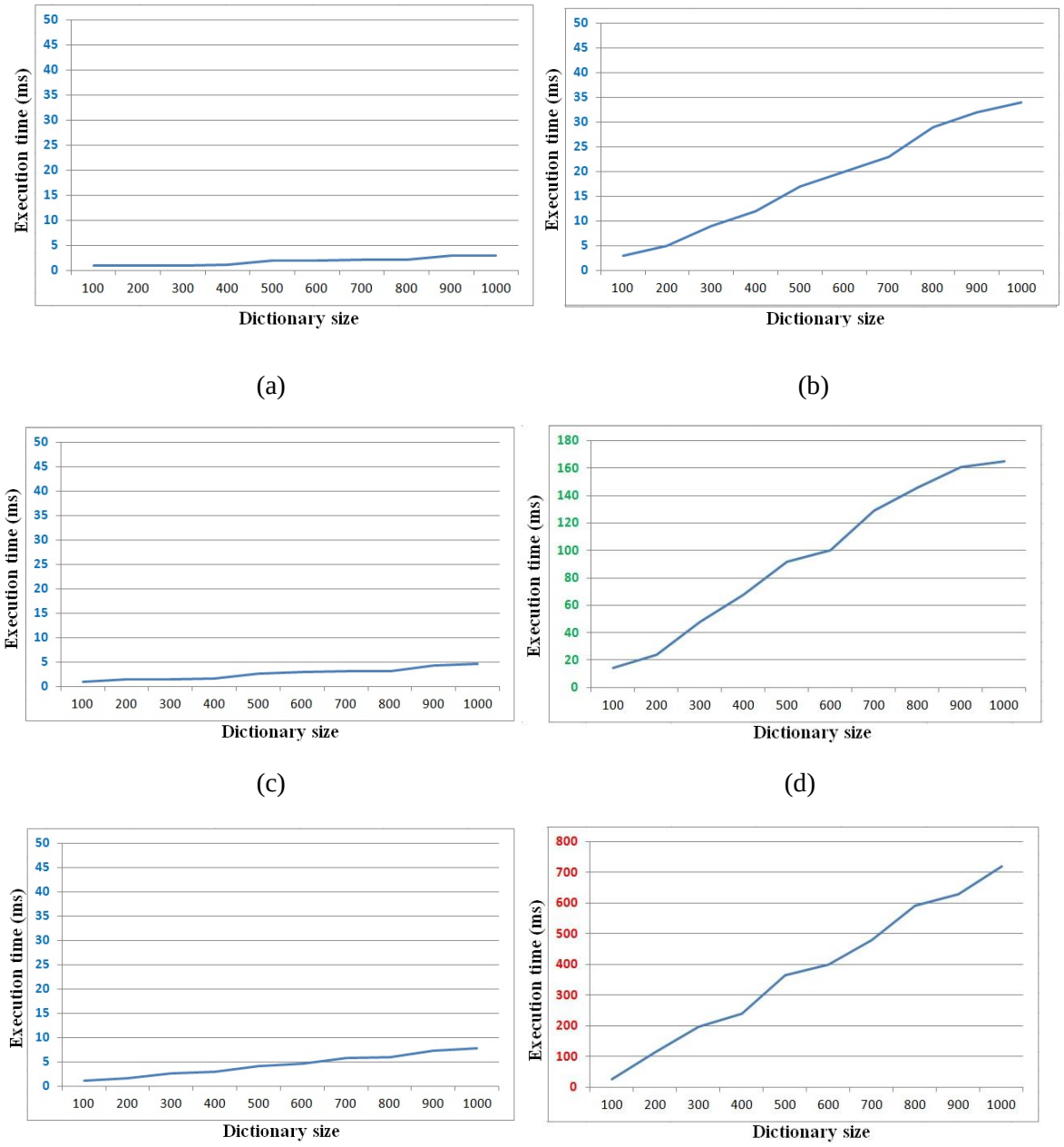


Figure.5.8 Temps d'exécution de la première étape de l'algorithme OMP. (a) Temps d'exécution pour un objet en parallèle. (b) Temps d'exécution pour un objet en séquentiel. (c) Temps d'exécution pour 5 objets en parallèle. (d) Temps d'exécution pour 5 objets en séquentiel. (e) Temps d'exécution pour 10 objets en parallèle. (f) Temps d'exécution pour 10 objet en séquentiel.

Tableau. 6: Le temps d'exécution sur la carte graphique

	1 objet			5 objets			10 objets		
Taille dictionnaire	GPU (ms)	CPU (ms)	Ratio (CPU/GPU)	GPU (ms)	CPU (ms)	Ratio (CPU/GPU)	GPU (ms)	CPU (ms)	Ratio (CPU/GPU)
100	0,9	3	3,33	1	14	14	1,08	26	24,07
200	0,93	5	5,37	1,41	24	17,02	1,56	115	73,71
300	0,98	9	9,18	1,54	48	31,16	2,62	197	75,19
400	1,06	12	11,32	1,58	68	43,03	3,04	240	78,94
500	1,92	17	8,85	2,63	92	34,98	4,21	366	86,93
600	1,94	20	10,30	2,99	100	33,44	4,59	400	87,14
700	2,06	23	11,16	3,12	129	41,34	5,83	480	82,33
800	2,14	29	13,55	3,14	146	46,49	6,03	591	98,00
900	2,89	32	11,07	4,25	161	37,88	7,27	630	86,65
1000	2,92	34	11,64	4,62	165	35,71	7,75	720	92,90

Tableau 7: Comparaison du nombre d'images traitées par seconde entre plusieurs méthodes

Méthode	**Approche proposée**	[Wu 05]	[Ben 11]	[Breitenstein 11]	[Possegger 14]
Temps (fps)	10-22	0.4-2	0,8	1-2	2,1

Nous remarquons d'après la figure 5.8 et le tableau 6 que le temps d'exécution de l'implémentation séquentielle augmente considérablement lorsque la taille du dictionnaire augmente ainsi que le nombre d'objets détectés. Concernant l'approche parallèle proposée adaptée pour le suivi d'objet, le gain est remarquable, il peut atteindre une accélération $\times 100$. Cette implémentation parallèle a permis d'assurer un système de suivi multi-objets à temps réel, plus performant par rapport à des travaux de l'état de l'art. Le tableau 7 présente une comparaison entre le temps d'exécution de notre approche et d'autres de l'état de l'art. Le nombre d'images traitées par seconde dépend du nombre des objets détectés. Nous remarquons que pour notre approche, la performance varie entre 10 fps (frame per second) et 22 fps. Pour [Wu 05] et [Breitenstein 11] la performance varie aussi selon le nombre d'objets détectés.

V.4 Conclusion

Pour accélérer le traitement nous avons proposé dans ce travail d'utiliser les performances des GPUs pour une implémentation parallèle. Notre deuxième contribution, est une nouvelle implémentation de l'algorithme OMP adaptée pour un système de suivi d'objets. Des parties de cet algorithme (OMP) ayant comme objectifs de classifier les objets pour une exacte reconnaissance ont été accélérées jusqu'à cent fois grâce aux GPUs. Nous avons comparé nos résultats avec d'autres de l'état de l'art et nous concluons que l'approche proposée est très performante et est capable de traiter jusqu'a 22 images par seconde.

Conclusion générale

Dans cette thèse, nous avons présenté un système de détection et d'estimation de mouvement appliqué pour le suivi d'objet en mouvement dans une scène. Nous avons étudié des techniques de détection de mouvement déjà réalisées dans l'état de l'art, et nous avons testé quelques unes d'elles dans différents types de vidéo. Nous avons choisi la technique de soustraction de fond pour implémenter notre détecteur de mouvement. Après des opérations de filtrage nécessaires pour l'extraction de l'avant plan de l'arrière plan, nous avons proposé une méthode pour représenter chaque objet en mouvement une fois détecté avec son apparence et sa position. Cette représentation n'est pas très robuste par rapport à d'autres de l'état de l'art utilisant des caractéristiques plus distinctives. Cependant, nous avons augmenté la précision de notre système de suivi en se basant sur les représentations parcimonieuses pour la classification de chaque objet détecté. Parmi les différents algorithmes de représentation parcimonieuse, nous avons choisi un algorithme appelé "Orthogonal Matching Pursuit" pour avoir le vecteur de parcimonie nécessaire pour l'étape de la classification. Cette classification est très utile pour la reconnaissance et la distinction de ces objets détectés. Cette tâche permet une facilité de l'estimation de mouvement de chaque objet dans la scène observée.

L'étape de la classification se base sur un dictionnaire d'apprentissage en ligne, ce dernier contient toutes les informations qui décrivent les objets détectés. Cependant, une fois que la taille de ce dictionnaire devient très importante, l'algorithme utilisé devient plus long et le système ne peut fonctionner en temps-réel. Pour accélérer le traitement sans réduire la précision, on a eu recours à une implémentation parallèle. Les cartes graphiques (GPU) conçues pour accélérer le graphisme des jeux vidéo sont maintenant des machines massivement parallèles. Nous avons exploité ces performances et implémenter l'algorithme "Orthogonal matchinf pursuit" en langage CUDA dédié au GPU.

Nous avons testé notre approche sur des vidéos de référence très utilisées dans les travaux de l'état de l'art. Les résultats obtenus sont très prometteurs, que ce soit du point de vue précision ou de rapidité. De ce fait on envisage pour des travaux ultérieurs d'aborder un système de surveillance et de suivi d'objet en mouvement dans un environnement multi-acquisition.

Notations

Nous présentons ici les notations utilisées tout le long du document.

$\lceil x \rceil$	*partie entière*
$\langle x, y \rangle$	*produit scalaire de deux vecteurs*
$[\![x, y]\!]$	*corrélation entre deux vecteurs*
$argmax$	*retourne l'argument maximum*
$\|x\|_0$	*La norme zéro*
$\|x\|_1$	*La norme un*
$\|x\|_2$	*La norme deux*
$supp(v)$	*Le nombre d'élément non nul du vecteur V*

Liste des figures

Bibliographie

[Aach 95] T. Aach, A. Kaup. – Bayesian algorithms for change detection in image sequences using markov random fields. Signal Processing : Image Communication, 7(2) :147–160, 1995.

[Ayer 95] S. Ayer, H. Sawhney. – Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. Proc.

[Andriluka 08] Andriluka, M., Roth, S., & Schiele, B. 'People-tracking-by-detection and people- detection- by-tracking'. CVPR 2008. IEEE Conference on, pp.1-8.

[Aaftab 08] Aaftab Munshi : OpenCL Parallele Computing on the GPU. Pré sentation à SIGGRAPH2008, 2008. (Cité pages 4, 26, 160 et 165.)

[Alexandre 06] Alexandre Chariot, Renaud Keriven et Romain Brette : Simulation rapide de modèles de neurones impulsionnels sur carte graphique. In 1ère conférence francophone de neurosciences computationnelles, Pont-à-Mousson, Oct 2006.

[Amit 99] Y., Geman, D.: A computational model for visual selection. Neural Computation 11 (1999) 1691–1715

[Bercla 06] Bercla, J., Fleuret, F., & Fua, P. 'Robust people tracking with global trajectory optimization'. In Computer Vision and Pattern Recognition, IEEE Computer Society Conference on. 2006. Vol. 1, pp. 744-750.

[Bouthemy 93] P. Bouthemy, P. Lalande. – Recovery of moving object masks in an image sequence using local spatiotemporal contextual information. Optical Engineering, 32(6) :1205–1212, June 1993.

[Ben 11] Ben Shitrit, H., Berclaz, J., Fleuret, F., & Fua, P. 'Tracking multiple people under global appearance constraints'. In Computer Vision (ICCV), November

[Bay 08] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. 'Speeded-up robust features (SURF)'. Computer vision and image understanding, 2008, 110(3), 346-359.

[Bernardin 08] Bernardin, K., & Stiefelhagen, R. 'Evaluating multiple object tracking performance: the CLEAR MOT metrics'. Journal on Image and Video Processing, 2008, 1.

[Breitenstein 11] Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Van Gool, L.'Online multiperson tracking-by-detection from a single, uncalibrated camera'. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2011,33(9), 1820-1833.

[Bergen 90] J. Bergen, P. Burt, R. Hingorani, and S. Peleg, "Computing two motions from three frames," in 3rd Int. Conf. on Computer Vision, pp. 27–32, 1990.

[Bao 12] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2012, pp. 1830–1837.

[Buck 07] Ian Buck : GPU computing with NVidia CUDA. In SIGGRAPH '07 : ACM SIGGRAPH 2007 courses, page 6, New York, NY, USA, 2007. ACM. pp 26-160.

[Butler 03] D. Butler, S. Sridharan, and V. M. Bove Jr, "Real-time adaptive background segmentation," in IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 3, pp. 349–352, 2003.

[Burt 89] P. Burt, J. Bergen, R. Hingorani, R. Kolczynski, W. Lee, A. Leung, J. Lubin, and J. Shvaytser, "Object tracking with a moving camera; an application of dynamic motion analysis," in IEEE Workshop on Visual Motion, (Irvine,CA), 1989.

[BrookGPU] Stanford University Graphics Group : BrookGPU. http: //graphics.stanford.edu/projects/brookgpu/.

[Bowden 01] R. Bowden and P. Kaewtrakulpong, "An improved adaptive background mixture model for real-time tracking with shadow detection," in AVBS01, 2001.

[Bradski 98] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," Intel Tech Journal, pp. 1–15, 1998.

[Cavallaro 00] A. Cavallaro, T. Ebrahimi. – Video object extraction based on adaptive background and statistical change detection. in Proc. of SPIE VCIP, 2000.

[Csurka 99] G. Csurka, P. Bouthemy. – Direct identification of moving objects and back-ground from 2d motion models. Proc. Int. Conf. Computer Vision, 1999.Int. Conf. Computer Vision, 1995.

[Cavallaro 00] A. Cavallaro, T. Ebrahimi. – Video object extraction based on adaptive background and statistical change detection. in Proc. of SPIE VCIP, 2000.

[Chariot 09] Alexandre Chariot, "Quelques Applications de la Programmation des Processeurs Graphiques à la Simulation Neuronale et à la Vision par Ordinateur", 2009.

[Dufaux 95] F. Dufaux and F. Moscheni. Segmentation-based motion estimation for second generation video coding techniques. Technical report, MIT, Media Lab. and Swiss FIT, S.P. Lab, 1995.

[Dai 14] Dai, Y., He, D., Fang, Y., & Yang, L. 'Accelerating 2D orthogonal matching pursuit algorithm on GPU'. The Journal of Supercomputing, 2014, 69(3), 1363-1381.

[Dalal 05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2005.

[Dollar 09] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In BMVC, 2009.

[Doretto 03] G. Doretto, A. Chiuso, Y.N. Wu, S. Soatto. – Dynamic textures. Int. J. Computer Vision, 51(2) :91–109, 2003.

[Dimitrova 95] N. Dimitrova, F. Golshani. – Motion recovery for video content classification. ACM Trans. Inf. Syst., 13(4) :408–439, 1995.

[Darrel 91] T. Darrel, A. Pentland. – Robust estimation of a multi-layered motion representation. IEEE Workshop on Visual Motion, 1991.

[David 06] David Tarditi, Sidd Puri et Jose Oglesby : Accelerator : using data parallelism to program GPUs for general-purpose uses. SIGARCH Comput. Archit. News, 34(5):325–335, 2006.

[Donoho 06] D.L. Donoho, M. Elad, and V. Temylyakov. Stable recovery of sparse overcomplete representations in the presence of noise. IEEE Transactions on I.T., pages 6–18,2006.

[Dibos 05] F. Dibos, G. Pelletier, S. andKoepfler. – Real-time segmentation of moving objects in a video sequence by a contrario detection. Proc. Int. Conf. Image Processing, 2005.

[Elgammal 00] A. Elgammal, D. Harwood, L. Davis. – Non-parametric model for background subtraction. Proc. Europ. Conf. Computer Vision, 2000.

[Efron 03] B. Efron, T. Hastie, and R. Tibshirani, “Least angle regression,” Annals of Statistics, vol. 32, pp. 407–499,2004.

[Elbahri 15,1] Elbahri, M., KPALMA, K., TALEB, N., & EL-MEZOUAR, M. C. (2015). A Novel Object Position Coding for Multi-Object Tracking using Sparse Representation. International Journal of Image, Graphics and Signal Processing (IJIGSP), 7(8), 1.

[Elbahri 15,2] Elbahri, M., TALEB, N., KPALMA, Joseph RONSIN. "Parallel algorithm implementation for multi-object tracking and surveillance". IET Computer Vision, In press

[Fang 11] Fang, Y., Chen, L., Wu, J., & Huang, B. 'Gpu implementation of orthogonal matching pursuit for compressive sensing'. In Parallel and Distributed Systems (ICPADS), December 2011, pp. 1044-1047.

[Friedman 97] N. Friedman, S. Russell. – Image segmentation in video sequences : A probabilistic approach. Uncertainty in Artificial Intelligence, pp. 175–181,1997.

[Flynn 72] Michael J. Flynn. Some computer organizations and their effectiveness. IEEE Trans. Comput., vol. 21, pages 948–960, September 1972.

[Fleet 90] D. Fleet, A. Jepson. – Computation of component image velocity from local phase information. Int. J. Computer Vision, 5(1) :77–104, 1990.

[Fuchs 04] J-J. Fuchs. Recovery of exact sparse representations in the presence of bounded noise. INRIA, IRISA, 2004.

[Feng 14] Q. Feng , "Novel classification rule of two-phase test sample sparse representation" Optik 125 (2014) 5825–5832

[Felzenszwalb 10]P. Felzenszwalb, R. B. Grishick, D.McAllister, and D. Ramanan. Object detection with discriminatively trained part based models. TPAMI, 32:1627–1645, 2010.

[Fuentes 03] L. M. Fuentes and S. A. Velastin, “Tracking people for automatic surveillance applications,” in Pattern recognition and image analysis (F.O. Perales,ed.), (Puerto de Andratx, Spain), pp. 238–245, Berlin; Springer;, 2003.

[Fuentes 01] L. M. Fuentes and S. Velastin, “People tracking in surveillance applications,” 2nd IEEE InternationalWorkshop on Performance Evaluation of Tracking and Surveillance (PETS2001), 2001.

[Fukunaga 90] K. Fukunaga, Introduction to Statistical Pattern Recognition. Boston: Academic Press, 1990.

[Grama 03] Ananth Grama, Anshul Gupta, George Karypis et Vipin Kumar. Introduction to parallel computing (second edition). Addison Wesley, 2003.

[Grimson 98] Y. Grimson, C. Stauffer, R. Romano, L. Lee. – Using adaptive tracking to classify and monitor activities in a site. Proc. Conf. Comp. Vision Pattern Rec.,1998.

[Gryn 05] J. Gryn, R. Wildes, J. Tsotsos. – Detecting motion patterns via direction maps with application to surveillance. Proc. of the Seventh IEEE Workshops on Application of Comp. Vision, 1 :202–209, 2005.

[Gordon 99] G. Gordon, T. Darrell, M. Harville, and J. Woodfill, “Background estimation and removal based on range and color,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 459–464, 1999.

[Geman 84] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of image. IEEE PAMI, 6(6) :721{741, November 1984.

[Hu 04] Weiming Hu, Tieniu Tan, Liang Wang, and S. Maybank. A Survey on Visual Surveillance of Object Motion and Behaviors. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 34(3):334–352, August 2004.

[Hu 2 04] M. Hu, W. Hu, and T. Tan, “Tracking people through occlusions,” in Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, pp. 724– 727, 2004.

[Huo 12] F. Huo, E.A. Hendriks, 'Multiple people tracking and pose estimation with occlusion estimation', Computer Vision and Image Understanding, 2012, 116, 634–647.

[Huang 08] Huang, C., Wu, B., & Nevatia, R. 'Robust object tracking by hierarchical association of detection responses'. In Computer Vision–ECCV, 2008, pp. 788-801. Springer Berlin Heidelberg.

[Hsu 84] Y. Hsu, H. Nagel, G. Rekers. – New likelihood test methods for change detection in image sequences. Comput. Vision, Graphics, Image Proc., 26(1) :73–106, 1984.

[Harris 88] C. Harris, M. Stephens. – A combined corner and edge detector. Alvey Vision Conference, 1988.

[Huwer 00] S.Huwer, H.Niemann.–Adaptive change detection for real-time surveillance applications.–Third IEEE International Workshop on Visual Surveillance,Dublin, 2000.

[Hee 87] D.J. Heeger. Optical flow using spatiotemporal filters. International Journal of Computer Vision, 1 :279,302, 1987.

[Heeger 88] D. Heeger. – Optical flow using spatio-temporal filters. Int. J. Computer Vision,1(4) :279–302, 1988.

[Horn 81] B. Horn, B. Schunck. – Determining optical flow. Artif. Intell., 17(1-3) :185–203, 1981.

[Haritaoglu 99] I. Haritaoglu, D. Harwood, and L. S. Davis, "Hydra: multiple people detection and tracking using silhouettes," in International Conference on Image Analysis and Processing, pp. 280 – 285, 1999.

[Haritaoglu 98] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who? when? where? what? a real time system for detecting and tracking people," in Third IEEE International Conference on Automatic Face and Gesture Recognition, pp. 222 –227, 1998.

[Haritaoglu 00]I. Haritaoglu, D. Harwood, and L. Davis, "An appearance-based body model for multiple people tracking," in 15th International Conference on Pattern Recognition, vol. 4, (Barcelona, Spain), pp. 184–187, 2000.

[Irani 98] M. Irani, P. Anandan. – A unified approach to moving object detection in 2d and 3d scenes. IEEE Trans. Pattern Anal. Machine Intell., 20(6) :577–589,1998.

[Horprasert 99] T. Horprasert, D. Hanvood, and L. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in ICCV Frame-rate Workshop, 1999.

[Harville 04] M. Harville and D. Li, "Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera," in Computer Vision and Pattern Recognition, 2004. CVPR 2004.

Proceedings of the 2004 IEEE Computer Society Conference on, pp. II–398– II–405 Vol.2, 2004.

[Hayman 03] E. Hayman and J. Eklundh, "Statistical background subtraction for a mobile observer," in International Conference on Computer Vision (ICCV), vol. 1,pp. 67 – 74, 2003.

[Jepson 93] A. Jepson, M. Black. – Mixture models for optical flow computation. Proc. Conf. Comp. Vision Pattern Rec., 1993.

[Jia 12] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2012, pp. 1822–1829.

[Javed 02] O. Javed, K. Shafique, and M. Shah, "A hierarchical approach to robust background subtraction using color and gradient information," in in IEEE Workshop on Motion and Video Computing, pp. 22–27, 2002.

[Karmann 90] K. Karmann, A. Brand. – Time-varying image processing and moving object recognition. – Elsevier Science Publish., 1990.

[Kim 05] K. Kim, D. Harwood, L. Davis. – Background updating for visual surveillance. Int. Symposium on Visual Computing., 2005.

[Koller 94] D. Koller, J. Weber, J. Malik. – Robust multiple car tracking with occlusion reasoning. – Proc. Europ. Conf. Computer Vision, 1994.

[Koller 94] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in ECCV, vol. 1, pp. 189–196, 1994.

[Konrad 00] J. Konrad. – Handbook of image and Video processing. – Academic press, 207–225p., 2000.

[Kon 00] KON F., ROMÁN M., LIU P., MAO J., YAMANE T., MAGALHÃES L. C., CAMPBELL R. H., « Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB », Proceedings of International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000), no 1795 LNCS, New York, April 2000, Springer-Verlag.

[Kang 03] S. Kang, B.-W. Hwang, and S.-W. Lee, "Multiple people tracking based on temporal color feature," International Journal of Pattern Recognition and Artificial Intelligence, vol. 17, no. 6, pp. 931–949, 2003.

[Kanade 98] T. Kanade, R. Collins, A. Lipton, P. Burt, L. Wixson. – Advances in cooperative multi-sensor video surveillance, 1998.

[Lei 05] B. Lei and L.-Q. Xu, "From pixels to objects and trajectories: a generic real-time outdoor video surveillance system," in Imaging for Crime Detection and Prevention, 2005. ICDP 2005. The IEE International Symposium on, pp. 117–122, 2005.

[Laptev 05] I. Laptev, S. Belongie, P. Pérez, J. Wills. – Periodic motion detection and segmentation via approximate sequence alignment. Proc. Int. Conf. Computer Vision, 2005.

[Lindholm 08] Lindholm, E., Nickolls, J., Oberman, S., & Montrym, J. 'NVIDIA Tesla: A unified graphics and computing architecture'. 2008, IEEE micro, (2), 39-55.

[Lisani 03] J. Lisani, J. Morel. – Detection of major changes in satellite images. Proc. Int.Conf. Image Processing, 2003.

[Lu 13] W. Lu, C. Bai, K. Kpalma and J. Ronsin "Multi-object Tracking using Sparse Representation", ICASSP 2013, May 26-31, 2013, Vancouver, Canada (2013).

[Lu 01] W. Lu and Y.-P. Tan, "A color histogram based people tracking system," in 2001 IEEE International Symposium on Circuits and Systems, vol. 2, pp. 137 – 140, 2001.

[Lowe 99] D.G. Lowe. – Object recognition from local scale-invariant features. Proc. Int. Conf. Computer Vision, 1999.

[Lucas 81] B.D. Lucas, T. Kanade. – An iterative technique of image registration and its application to stereo. Proc. Int. Joint Conf. on Artificial Intelligence, 1981.

[Labatut 06] P. Labatut, R. Keriven et J.P. Pons : A GPU Implementation ofLevel Set Multiview Stereo. International Conference on Computational Science (4) pp, pages 212–219, 2006.

[Matsumura 02] A. Matsumura, Y. Iwai, and M. Yachida, "Tracking people by using color information from omnidirectional images," in 41st SICE Annual Conference,vol. 3, pp. 1772 – 1777, 2002.

[Mark 05] Mark J. Harris : Mapping computational concepts to GPUs. In SIGGRAPH '05 : ACM SIGGRAPH 2005 Courses, page 50, New York, NY, USA, 2005. ACM.

[Maji 08] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel machines is efficient. In CVPR,2008.

[Mei 11] X. Mei and H. Ling, "Robust visual tracking and vehicle classification via sparse representation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 11, pp. 2259–2272, Nov. 2011.

[Mikolajczyk 07] K. Mikolajczyk et J. Matas : Improving Descriptors for Fast Tree Matching by Optimal Linear Projection. Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–8, 2007.

[Martin 10] Aurélie Martin "Représentations parcimonieuses adaptées à la compression d'images", 2010.

[Mohan 01] Mohan, A., Papageorgiou, C., Poggio, T.: Example-based object detection in images by components. IEEE Trans. on Pattern Analysis and Machine Intelligence 23 (2001) 349–361

[Nelson 91] R. Nelson. – Qualitative detection of motion by a moving observer. Proc. Conf.Comp. Vision Pattern Rec., 1991.

[Neil 08] Neil Trevett : OpenCL - Heterogeneous Parallel Programming. SIGGRAPH'08 BOF slides, 2008.

[Nicolas 04] Nicolas Fritz, Philipp Lucas et Philipp Slusallek : CGiS, a new Language for Data-Parallel GPU Programming. In Bernd Girod, Hans-Peter Seidel et Marcus Magnor, éditeurs : Proceedings of the 9th International Workshop "Vision, Modeling, and Visualization" (VMV'04),2004.

[NP 02] K. P. Ng and S. Ranganath, "Tracking people," in 16th International Conference on Pattern Recognition, vol. 2, pp. 370 – 373, 2002.

[Odobez 97] J.-M. Odobez, P. Bouthemy. – Separation of moving regions from background in an image sequence acquired with a mobile camera. – Kluwer Academic Publisher, 283–311p., 1997.

[Pundlik 06] S. Pundlik, S. Birchfield. – Motion segmentation at any speed. Proc. of the British Machine Vision Conf., 2006.

[Possegger 14] Possegger, H., Mauthner, T., Roth, P. M., & Bischof, H. 'Occlusion geodesics for online multi-object tracking'. In Computer Vision and Pattern Recognition (CVPR), June 2014 IEEE Conference on, pp. 1306-1313.

[Patrick 04] Patrick S. McCormick, Jeff Inman, James P. Ahrens, Charles Hansen et Greg Roth : Scout : A Hardware-Accelerated System for Quantitatively Driven Visualization and Analysis. In VIS '04 : Proceedings of the conference on Visualization, 2004.

[Radke 05] R.J. Radke, S. Andra, O. Al-Kofahi, B. Roysam. – Image change detection algorithms : a systematic survey. IEEE Transactions on Image Processing, 14(3) :294–307, 2005.

[Rao 00] A. Rao, R. Srihari, and Z. Zhang, "Geometric histogram: A distribution of geometric configurations of color subsets," in SPIE: Internet Imaging, vol. 3964, pp. 91–101, 2000.

[Rosin 98] P. Rosin. – Thresholding for change detection. Proc. Int. Conf. Computer Vision, pp. 274–279, 1998.

[Roth 02] Roth, D., Yang, M.H., Ahuja, N.: Learning to recognize 3d objects. Neural Computation 14 (2002).

[Rigoll 00] G. Rigoll, S. Eickeler, and S. Muller, "Person tracking in real-world scenarios using statistical methods," in Automatic face and gesture recognition, (Grenoble, France), pp. 342–347, IEEE; 2000, 2000.

[Sahouria 97] E. Sahouria, A. Zakhor. – Motion indexing of video. Proc. Int. Conf. Image Processing, 1997.

[Shi 94] J. Shi, C. Tomasi. – Good features to track. Proc. Conf. Comp. Vision Pattern Rec., 1994.

[Spinei 98] A. Spinei, D. Pellerin, J. Herault. – Spatiotemporal energy-based method for velocity estimation. Signal Processing, 65 :347–362, 1998.

[Stauffer 99] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in IEEE Computer Society Conference on Computer constraint. IEEE Trans. Pattern Anal. Machine Intell., 15(2) :162–166,1993.

[Tomasi 91] C. Tomasi, T. Kanade. – Detection and Tracking of Point Features. – Rapport de Recherche nCMU-CS-91-132, Carnegie Mellon University, April 1991.

[Tian 05] Y.L. Tian, A. Hampapur. – Robust salient motion detection with complex background for real-time video surveillance. Workshop on Motion and Video Computing, 2005.

[Tropp 03] J.A. Tropp. Greed is good : algorithmic results for sparse approximation. IEEE Transactions of Information Theory, 2003.

[Tony 11] T. Tony Cai and Lie Wang "Orthogonal Matching Pursuit for Sparse Signal Recovery With Noise" IEEE TRANSACTIONS ON INFORMATION VOL. 57, NO. 7, JULY 2011.

[Toyama 99] K. Toyama, J. Krumm, B. Brumitt, B. Meyers. – Wallflower : Principles and practice of background maintenance. – Proc. Int. Conf. Computer Vision, 1999.

[Thuan 11] Hiep-Thuan DO, Extensibilité des moyens de traitements pour les données issues des vastes systèmes d'informations géographiques. 2011

[Thongkamwitoon 04]T. Thongkamwitoon, S. Aramvith, and T. Chalidabhongse, "An adaptivereal- time background subtraction and moving shadows detection," in IEEE International Conference on Multimedia and Expo (ICME), vol. 2, pp. 1459–1462, 2004.

[ULLMAN 79] ULLMAN, Shimon. The interpretation of visual motion. Massachusetts Inst of Technology Pr, 1979.

[Veit 05] T. Veit, F. Cao, P. Bouthemy. – A maximality principle applied to a contrario motion detection. Proc. Int. Conf. Image Processing, 2005.

[Vidal 04] R. Vidal, Y. Ma. – A unified algebraic approach to 2-d and 3-d motion segmentation. Proc. Europ. Conf. Computer Vision, 2004.

[Vidal 05] R. Vidal, D. Singaraju. – A closed form solution to direct motion segmentation. Proc. Conf. Comp. Vision Pattern Rec., 2005.

[Vedaldi 09] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In ICCV, 2009.

[Viola 01] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In:Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2001).

[Welch 95] G. Welch and G. Bishop, "An introduction to the kalman filter," technical report tr95-041, University of North Carolina at Chapel Hill, 1995.

[Wang 02] L. Wang, W. Hu, and T. Tan, "Face tracking using motion-guided dynamic template matching," ACCV'2002, 2002.

[Wren 97] C.R. Wren, A. Azarbayejani, T. Darrell, A. Pentland. – Pfinder : Real-time tracking of the human body. IEEE Trans. Pattern Anal. Machine Intell,19(7) :780–785, 1997.

[Wu 05] Q. Wu, H. Cheng, B. Jeng. – Motion detection via change-point detec tion for cumulative histograms of ratio images. Pattern Recognition Letters, 26(5) :555–563, 2005.

[Wu.B 05] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In ICCV, 2005.

[Wang 94] J. Y. A. Wang, E. H. Adelson. – Representing moving images with layers. IEEE Trans. on Image Processing Special Issue, 3(5) :625–638, 1994.

[Wang 05] H. Wang and D. Suter, "A re-evaluation of mixture-of-gaussian background modeling," in IEEE International Conference on Acoustics, Speech, and Signal Processing, (Philadelphia, PA, USA), pp. 1017–1020, 2005.

[Weiss 97] Y. Weiss. – Smoothness in layers : Motion segmentation using nonparametric mixture estimation. Proc. Conf. Comp. Vision Pattern Rec., 1997.

[Weber 00] Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: Proceedings of the Sixth European Conference on Computer Vision. (2000) 18–32

[Wildes 98] R. Wildes. – A measure of motion salience for surveillance applications. Proc.Int. Conf. Image Processing, 1998.

[Wixson 00] L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. IEEE Trans. Pattern Anal. Machine Intell., 22(8) :774–780, 2000.

[Wang 12] Q. Wang, F. Chen, W. Xu, and M.-H. Yang, "Online discriminative object tracking with local sparse representation," in Proc. IEEE Workshop Applicat. Comput. Vision, 2012, pp. 425–432.

[Wynn 04] C. Wynn : OpenGL Render-to-Texture. Nvidia Corporation, white paper Edition, 2004.

[Xiao 05] J. Xiao, M. Shah. – Accurate motion layer segmentation and matting. Proc.Conf. Comp. Vision Pattern Rec., 2005.

[Zhu 05] S. Zhu, Q. Avidan, K.-T. Cheng. – Learning a sparse, corner-based representation for time-varying background modeling. Proc. Int. Conf. Computer Vision,2005.

[Zhu 10] L. Zhu, Y. Chen, A. Yuille, and W. Freeman. Latent hierarchical structural learning for object detection. In CVPR, 2010.

[Zang 04] Q. Zang and R. Klette, "Robust background subtraction and maintenance," in Proceedings of the 17th International Conference on Pattern Recognition (ICPR), vol. 2, pp. 90–93, 2004.

[Zhang 14] J. Zhang , H. Zhang , Z. Li ,"A hierarchical structure with improved OMP sparse representation used with face recognition", Optik 125 (2014) 4729–4735 2011, IEE International Conference on, pp. 137-144.

[Zhang 12] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2012, pp. 2042–2049.

[Zhong 03] J. Zhong, S. Sclaroff. – Segmenting foreground objects from a dynamic textured background via a robust kalman filter. Proc. Int. Conf. Computer Vision,2003.

[Zhao 04] T. Zhao and R. Nevatia, "Tracking multiple humans in complex situations," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 9, pp. 1208–1221, 2004.