Université Bordeaux1

Laboratoire Bordelais de
Recherche en informatique

Mémoire de stage de Master2 Recherche

Résponsable : Serge CHAUMETTE

Equipe : Système et Objets Distribués

Localisation, visée et gestuelles sur téléphone avec caméra

Ruding LOU

Juin 2006

Remerciements

Je veux exprimer toute ma reconnaissance profonde à mon directeur du mémoire, Monsieur le professeur **Serge CHAUMETTE**, qui m'a aidé à trouver ce stage de recherche et qui a su me guider dans cette recherche pour élaborer ce travail.

Merci à tous mes camarades de classe qui m'ont donné la critique ou le conseil constructif me permettant d'enrichir la réflexion sur l'exploitation du sujet.

Je tiens également à remercier Monsieur René ANDRON pour m'avoir soutenu dans le travail de rédaction de ce mémoire en langue française.

Une pensée particulière va à ma mère qui m'a encouragé et soutenu moralement tout au long de cette mission de l'apprentissage de recherche.

Résumé

Dans ce mémoire nous avons travaillé sur la possibilité de la localisation et les gestuelles de téléphone avec caméra. Notre étude consiste aux méthodes d'analyser les images capturées par la caméra pour détecter le mouvement de téléphone. Nous avons cherché dans le domaine de traitement d'image, des méthodes qui peuvent servir à calculer le mouvement, et nous les avons étudiées pour voir comment ces méthodes peuvent servir à l'objectif de notre recherche. Les méthodes que nous avons adoptées dans cette recherche sont le suivi d'objet, la stabilisation d'image, la souris optique, le flot optique, et l'analyse de bloc.
Trois expérimentations que nous avons effectuées sont aussi présentées, elles sont l'implémentation de l'appariement de bloc, la simulation du fonctionnement de souris optique basé sur le suivi d'objet, l'implémentation d'un SourisPhone sur un téléphone portable.

Mots Clefs

Téléphone portable, caméra embarquée, localisation, mouvement.

Abstract

In this memory we have worked for the localization and gesticulation possibilities on a telephone with camera. That consists with the methods to analyze the captured images by camera for detect the telephone motion. We sought methods in the image processing field, which can be used to calculate the motion. And we studied them to find how we can use them for our objective. The methods which we adopted are the object tracking, the image stabilization, the optical mice, the optical flow and the bloc analysis.
Three experiments are presented in this memory. They are the bloc matching implementation, the optical mice function simulation based on the object tracking, and finally the SourisPhone implementation on mobile telephone.

Keywords

Mobile telephone, embedded camera, localization, motion

Table des matières

Table des matières

Introduction

Généralité

Le problème de la géolocalisation d'équipement mobile est largement étudié aujourd'hui. Les systèmes actuels se basent sur l'existence d'un système global ou d'un système local.

Les systèmes globaux utilisent des techniques de type GPS **(Global Position System)**. Les équipements reçoivent leur localisation depuis un ou plusieurs satellites. Les systèmes locaux utilisent des techniques radio comme par exemple Bluetooth. Plusieurs équipements communiquent afin de déterminer leurs coordonnées par des approches de type triangulation.

Le problème de la visée et de la reconnaissance de gestuelles à l'aide d'un équipement mobile portable est lui aussi étudié. Il s'agit ici de déterminer les mouvements (et de déduire des gestuelles ou des objet visés) par un pointeur. Ceci nécessite aujourd'hui d'embarquer des matériels spécifiques (gyroscopes par exemple) ou d'installer des points d'infrastructure (avec lesquelles on communique par Bluetooth).

Dans le cadre de ce mémoire nous nous proposons d'étudier la possibilité de fournir l'ensemble de ces fonctionnalités sur un équipement autonome qui ne peut donc reposer que sur lui-même et qui n'est pas doté d'un matériel spécifique à cette fonction. On suppose simplement que l'équipement, un téléphone par exemple, est équipé d'une caméra. En activant la caméra, en récupérant le flux d'images généré par celle-ci et en l'analysant, on reconstruit les mouvements du mobile. On peut alors déterminer sa position, les gestuelles réalisées ou les objets vers lesquels on le pointe.

Notre domaine

Le [22] présente un système de localisation à l'intérieur d'un bâtiment. Ce système détermine les positions de l'utilisateur en analysant les images capturées par la caméra sur la smartphone (téléphone portable, PDA, etc…). En fait l'utilisateur se

déplace avec le smartphone dans une zone bien définie et encadrée. Le smartphone a une caméra embarquée. La zone définie signifie qu'ils ont une base de donnée pour stocker toutes les images précapturées en différentes orientations et en différentes positions. La caméra capture les différentes images sur les différentes positions de l'utilisateur. Le smartephone ensuite envoie les images au serveur par GPRS (GeneralPacketRadioService). Le serveur fait l'analyse des images reçues. Pour cela le serveur apparie l'image reçue avec les images précapturées dans la base de données pour trouver une image qui est la plus similaire de l'image reçue dans la base. Puis il identifie la position correspondante de l'image trouvée dans la base. Enfin il renvoie la position au smartphone par GPRS. Ainsi la localisation est réussie. Au niveau de la technique de comparaison d'image, elle utilise la technique d'histogramme de couleur, de décomposition d'ondelettes et d'appariement de formes.

Ce projet présenté ci-dessus est différent de ce qui est décrit dans ce mémoire. Notre objectif est de détecter le mouvement ou la gestuelle de smartphone. Ceci veut dire que l'utilisateur tenant un smarphone avec une caméra activée se déplace, et on analyse les images capturées pour détecter le mouvement du smartphone entre deux images adjacentes. C'est une technique basée sur une comparaison de deux images adjacentes. On n'a pas besoin de préconnaître le site ni de préparer la base de données, l'utilisateur est libre de se déplacer. Par contre le [22] a besoin de connaître le site d'abord et l'utilisateur ne doit pas déborder le site défini, mais la localisation est plus précise. De ce fait, le [22] présente une localisation absolue, et notre mémoire, une localisation relative.

Précision de la problématique

En fait ce sujet est inspiré de la souris optique. Le mécanisme de localisation de la souris optique est une localisation relative, car elle peut fonctionner sur n'importe quel plan, et elle se localise par l'analyse du mouvement. Mais quand on utilise la souris, elle effectue une localisation absolue sur l'écran. De là, nous avons une idée à faire fonctionner le téléphone comme une souris optique. Au début de la vie de souris optique, elle avait besoin d'un tapis spécial pour qu'elle puisse détecter son propre mouvement. L'idée est de détecter le mouvement de téléphone en mouvement devant un "tapis" spécial. Par exemple si l'on fait une croix avec le téléphone à la main, l'ordinateur peut reproduire la trajectoire de cette croix.

Si cette idée est bien fondée, on pourra envisager à agrandir le "tapis" ou à généraliser le tapis à une vue normale. Par exemple cette vue normale peut être un côté d'une pièce, elle contient des objets différents. On pourra donc se déplacer dans un domaine plus large en tenant la caméra et le mouvement de la caméra est détecté par la localisation relative, alors le mouvement de l'utilisateur est donc détecté. Si on veut aller plus loin, on pourrait tourner le téléphone au lieu de se déplacer en face d'une vue de 2-D. Donc le "tapis" d'ici n'est plus un plan, mais en 3-D cubique, composé par plusieurs plans 2-D. Ainsi on peut déduire que notre objectif est de mener une recherche sur le SourisPhone 3-D.

Pour faire cette recherche, il faut envisager des études sur le traitement

d’image pour détecter le mouvement. Plusieurs études existantes peuvent servir à notre recherche.

1. La méthode de suivi d’objet est largement étudiée et utilisée dans les domaines de surveillance de véhicule et d’intelligence artificielle. Alors notre idée est qu’au lieu de suivre l’objet, on va se suivre inversement.

2. Nous nous intéressons aux études de stabilisation d’image. Dans ce domaine on traite les images qui sont floues à cause de mouvement propre de caméra. On essaie d’enlever les flous en faisant une compensation. Cela consiste à calculer l’orientation vers laquelle l’image peut être bien compensée. Cette orientation dépend du mouvement de caméra.

3. Nous allons présenter la souris optique, pour voir si le mécanisme de la localisation de souris optique peut servir à cette recherche.

4. Le flot optique est présenté à la suite. Le flot optique est le mouvement apparent sur l’image. La technique de calcule du flot optique pourras donc nous aider pour calculer le mouvement de caméra.

5. Nous allons présenter l’étude de l’analyse de bloc qui consiste à la recherche de la partie commune dans deux images afin de trouver le vecteur de mouvement entre deux images.

A la suite de la présentation de ces études, sont présentés l’implémentation que nous avons réalisée pour faire le prototypage de cette recherche, le programme développé pour calculer le vecteur de mouvement entre deux images en s’appuyant sur la méthode d’appariement de bloc, ainsi que l’implémentation sur téléphone pour simuler le fonctionnement de la souris optique.

Chapitre 1

Le Suivi d'Objet

Dans ce chapitre on va étudier la méthode qui inspire ce sujet. C'est le " suivi d'objet " qui sert à surveiller l'objet bougeant dans un champ. C'est une étude qui existe, et pour nous au lieu de suivre l'objet, on va se suivre. Dans le champ de notre caméra, quand la caméra bouge, les objets fixes bougent aussi en sens inverse par rapport à la caméra. Donc si on peut calculer le mouvement d'objet dans le champ de la caméra, on peut déduire le mouvement de la caméra.

Il existe une méthode dans le " suivi d'objet ": la soustraction de l'arrière-plan. Elle fonctionne comme ceci : On dispose d'une ou plusieurs caméras fixes, et si un objet passe dans le ou les champs des caméras, la station peut suivre cet objet. L'idée est que d'abord toutes les caméras capturent le fond de l'image sans aucun objet en mouvement, c'est la notion d'***image_background***. Et ensuite le processus commence, chaque caméra capture les images par le biais d'une fréquence pertinente, ces images sont nommées ***image_surveillance*** . Donc le PC fait une soustraction, où l'on a : soustraction = ***image_background*** − ***image_surveillance***. S'il n'y a aucun objet en mouvement en avant plan, la soustraction est normalement nulle. S'il y a un objet en mouvement dans le champ, la soustraction donne une image noire sauf pour la partie où il y a l'objet. Donc si l'objet se déplace dans l'endroit où au moins une caméra le capture, on peut le suivre.

En fait c'est une méthode bien encadrée qui permet de suivre des objets. Cela veut dire que chaque caméra est fixe, et elles ont leur champ constant. Donc l'arrière-plan ou la partie de la vue sans l'objet pour la caméra est toujours la même. Cela représente une contrainte majeure, parce que notre recherche est de détecter le mouvement de la caméra, donc la caméra n'est pas fixe ainsi que le champ de la caméra change aussi toujours. De ce fait, si on veut suivre un objet en mouvement dans le champ de la caméra, cette méthode ne peut pas très bien fonctionner. Bien sûr il existe des études beaucoup plus sophistiquées qui analysent les images capturées par la caméra en mouvement de suivre des objets, mais l'objectif de notre recherche n'est pas de suivre les objets, mais de détecter le mouvement de la caméra grâce aux méthodes de suivi l'objet.

Alors pour prouver la viabilité de cette idée, on peut même utiliser la méthode encadrée expliquée au-dessus avec la caméra fixe. Normalement quand on bouge la caméra, l'arrière-plan de la caméra se modifie presque obligatoirement. Mais supposons que l'on ait fait en sorte d'avoir un arrière-plan invariant, donc l'optique de la caméra envisage une partie d'image identique (arrière-plan) et une autre partie complètement différente et isolée. Pour illustrer cette idée, imaginons par exemple une grande feuille blanche avec un carreau de couleur très vive. On appelle ceci une ***feuille à carreaux***. La taille de ce carreau est suffisamment réduite pour qu'il puisse bouger avec une sensibilité prédéfinie. On va donc pouvoir déplacer la caméra tel que le carreau soit présent dans les différentes positions, l'arrière-plan reste invariable. Au final on aura différentes positions du carreau en fonction de la position de la caméra, et l'arrière-plan reste constant, blanc. On peut utiliser la méthode de " suivi d'objet "ci-dessus pour repérer les différentes positions du carreau dans les images de la caméra et en déduire le mouvement de la caméra.

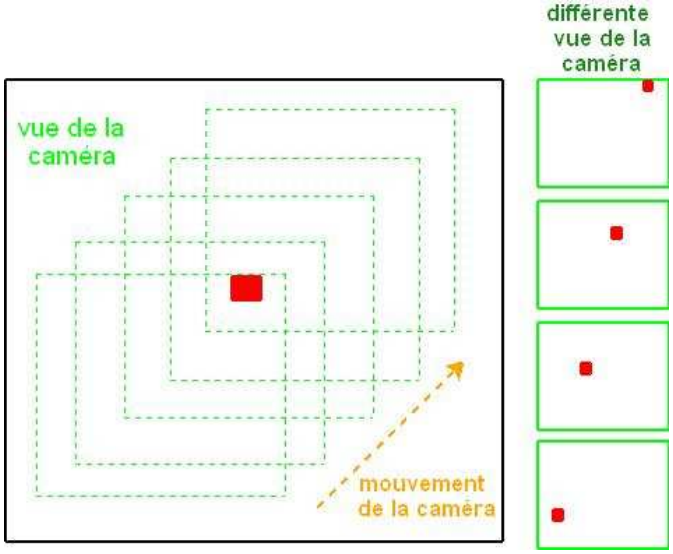

Figure 1-1 *Feuille à carreaux*

Une fois que tout cela est fait, on peut continuer à dériver des cas plus compliqués et moins encadrés. Dans l'exemple ci-dessus, la largeur du mouvement de la caméra est bien définie de telle sorte que le carreau soit toujours capturé par la caméra, donc le mouvement de la caméra est très limité. Pour faire évoluer ce modèle

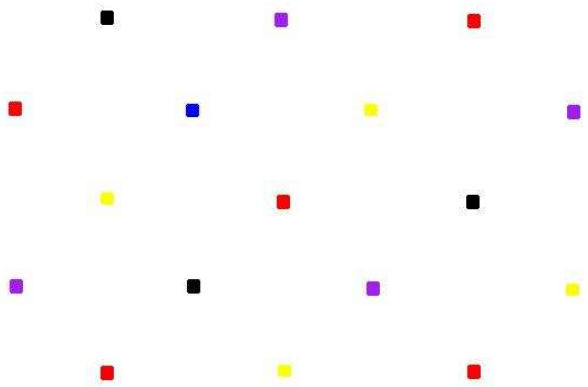

Figure 1-2 *Un plan contenant plusieurs particules*

on peut imaginer une vue qui contient un ensemble des sous-feuilles à carreaux, et sur chaque sous feuille à carreaux, la couleur du carreau est différente que celles des feuilles de carreau adjacentes. On peut appeler les carreaux les ***particule***. La Figure1-2 ci-dessus montre le plan qui contient plusieurs rayons de figure 1-1. Tous les carreaux de différentes couleurs adjacentes sont les ***particules***.

Dans ce cas les distances entre chaque deux particules voisines doivent être d'une longueur pertinente. Cela veut dire que d'abord la caméra doit capturer au moins une particule. Donc il faut éviter le cas dans la Figure 1- 3.

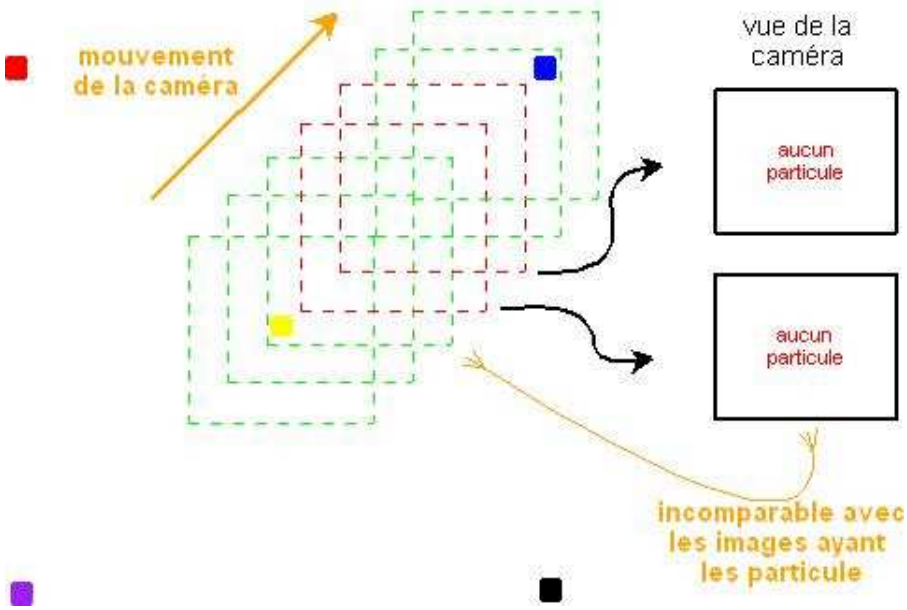

Figure 1-3 *La distance entre chaque deux particules voisines est trop grande*

Les distances ne peuvent pas être trop grandes, comme le montre la figure 1-3. Mais elles ne doivent pas être trop petites non plus. Voir la figure 1-4, on peut définir une ***boule*** d'une ***particule centrale*** et ses voisines. Donc il n'y aura qu'une boule qui apparaît dans le champ de la caméra.

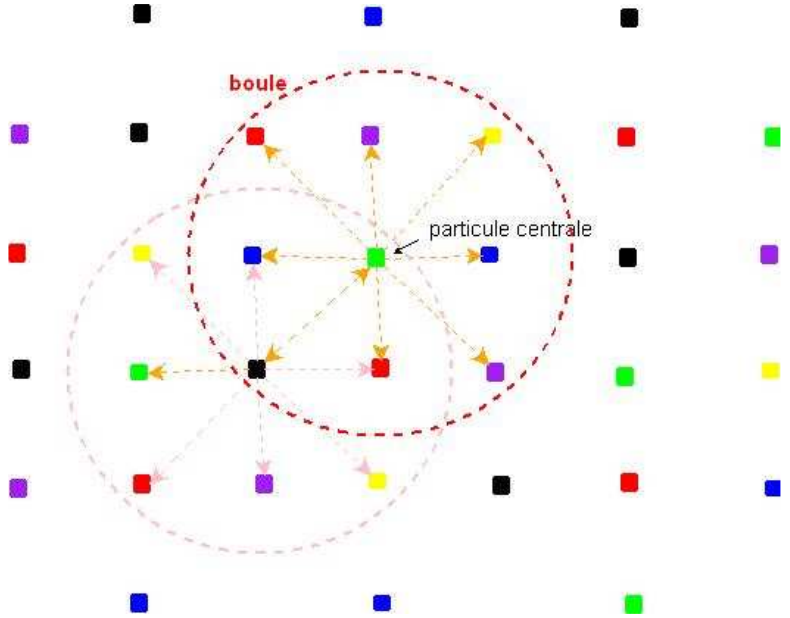

Figure 1-4 *Définition de boule*

Ensuite On se base sur une particule centrale pour calculer son mouvement dans le champ de la caméra. Une fois que cette particule n'apparaît plus dans le champ de la caméra, on va choisir dans cette vue une de ses particules voisines, elle doit apparaître aussi dans la vue précédente. Si la caméra capture plusieurs boules en même temps, donc il y aura donc des particules ayant la même couleur.

La Figure 1-6 illustre un plan dérivé de la feuille à carreaux de Figure 1-1, l'intérêt de ce plan est que l'on peut continuer bouger la caméra même si la particule suivie n'apparaît plus dans le champ de la caméra. C'est le ***calcul global***. La durée où la particule choisie apparaît toujours est la période de ***calcul local***, cela peut être considéré comme le cas d'une grande feuille blanche ayant une particule.

La figure 1-5 illustre les images capturées par la caméra qui a un mouvement illustré dans la Figure 1-6.

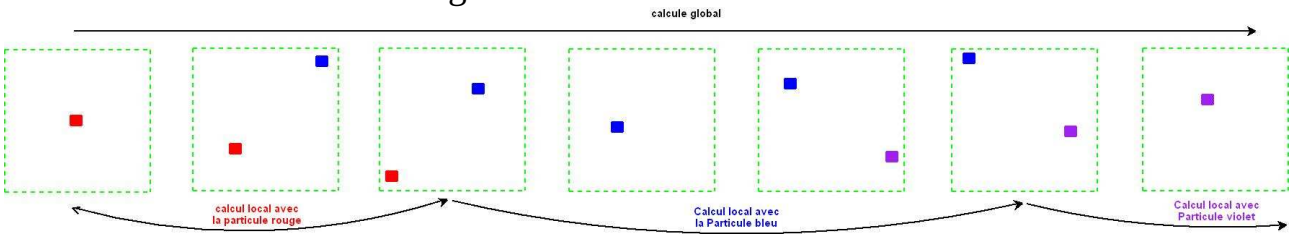

Figure 1-5 *Une suite de vues capturées par la caméra*

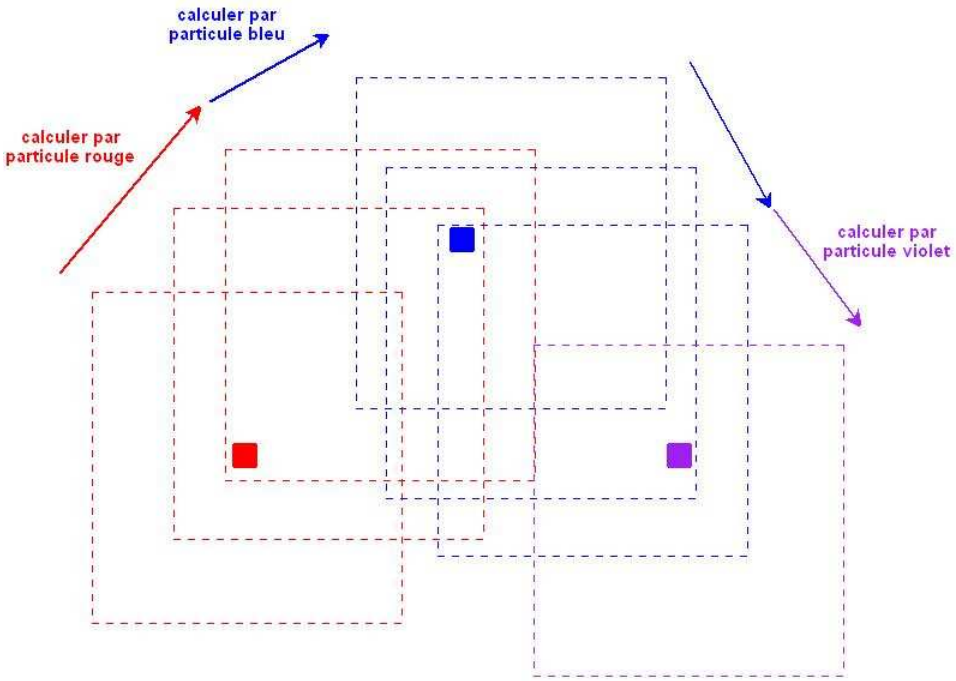

Figure 1-6 *La caméra se déplace devant une grande feuille ayant plusieurs particules*

En calculant son mouvement on peut restituer la trajectoire de ce mouvement.

Exemple 1-1

Il existe plusieurs méthodes de Suivi d'Objet. Par exemple sur l'article [3] « ***A Fast Model-Free Morphology-Based Object Tracking Algorithm*** », on a la description d'une méthode qui détecte l'objet par différenciation de l'arrière-plan. C'est la méthode la plus utilisée dans la plupart des algorithmes de suivi d'objet pour segmenter l'objet en mouvement. Les auteurs ont choisi la méthode de low-pass filter, parce que la méthode basée sur le calcul de flot optique demande un très gros investissement en calcul. Pour avoir un générateur d'arrière-plan assez calculable en temps réel et pour ne pas dépendre des matériels spéciaux, la méthode de low-pass filter est plus adéquate. La méthode de low-pass filter est capable de traiter les cas ayant des changements de luminance lents, comme l'ombre générée par des objets statiques. A l'instant ***t***, la différence entre arrière-plan et image courante est calculée par la formule:

$$\delta(r,c) = \begin{cases} 1 & |I(r,c) - B(r,c)| > T \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

où le ***r*** et ***c*** sont numéros de ligne et de colonne du pixel, ***δ*** est l'image de différence, ***I*** est l'image entrée, et ***B*** est l'image de l'arrière-plan. Donc si la différence ***δ(r, c)*** est supérieure au seuil ***T***, le pixel est appelé avant-plan. En même temps l'arrière-plan est mis à jour par l'expression 2,

$$B(t) = \beta \times I(t) + (1-\beta) \times B(t-1) \quad \text{if} \quad \delta(r,c) = 0 \qquad (2)$$

où ***β*** = 0.1 et il contrôle le taux de mis à jour de l'arrière-plan. Cette étape est la première étape où on a l'image de différenciation, donc tous les objets entrants peuvent être présentés par les silhouettes. Si l'on prend un exemple d'un piéton, ses silhouettes segmentées sont présentées dans la Figure 1-7.

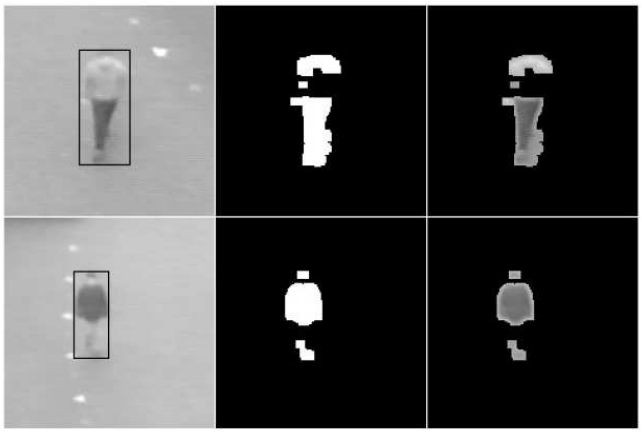

Figure 1-7 *Exemple d'objet à silhouette*

Le suivi d'objet décrit ici est l'appariement basé sur la notation "**d'objet à silhouette**". Pour identifier les différentes silhouettes de différents objets, on associe un profil à chaque silhouette. Le profil est un vecteur, $\mathbf{f_i}$ = [a,w,h,g] où ***g*** est un vecteur de 16 éléments pour l'histogramme de niveaux de gris associé par silhouette ***i*** (figure 1-8 c,d) , et ***a*** est la surface où le nombre de pixels contenus par la silhouette, ***w*** et ***h*** sont la largeur et la hauteur de rectangle minimum encadrant la silhouette.

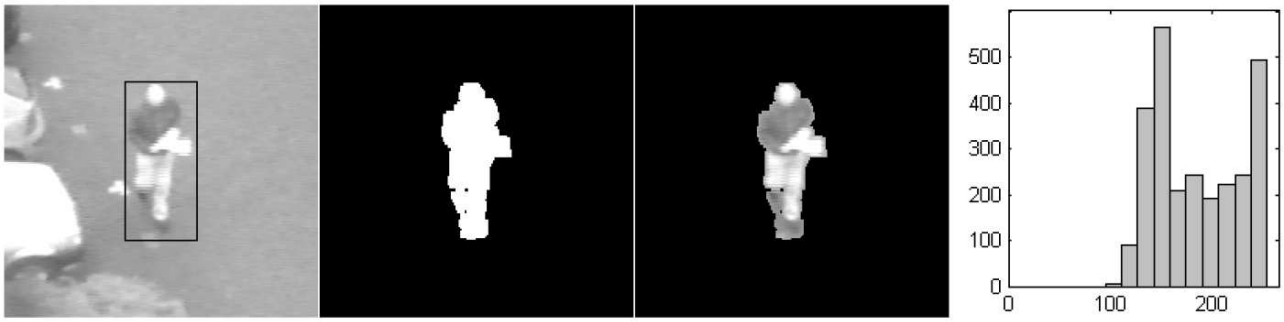

Figure 1-8 ***a*** *est l'objet encadré par un rectangle minimum calculé par silhouette dans* ***b;*** ***d*** *est l'histogramme de niveau de gris pour segmenté objet* ***c*** *)*

Le principe de l'algorithme ci-dessous est basé sur la "différence" entre les vecteurs de profils de l'objet et de la silhouette. Donc la différence entre les vecteurs de profil de l'objet ***Q*** et de la silhouette ***S*** est définie comme:

$$\mathbf{d}(Q,S)=\left[\left|a_Q-a_S\right|,\left|h_Q-h_S\right|,\left|w_Q-w_S\right|,\sqrt{\left(\mathbf{g}_Q-\mathbf{g}_S\right)\bullet\left(\mathbf{g}_Q-\mathbf{g}_S\right)}\right] \qquad (3)$$

C'est un vecteur de 4 éléments, il comprend entre l'objet et la silhouette, les quatre différences absolues au niveau de : la surface, l'hauteur, la largeur, la longueur scalaire de l'histogramme de niveau de gris.

Étape 1. Appariement Naïf : Les distances entre le centre de gravité de chaque objet ***q*** et le centre de gravité de chaque silhouette ***s,*** sont utilisées pour construire une matrice d'appariement valide, ***V*** , elle est basée sur un rayon autour de ***q***. Le rayon ***r*** = 80 est pour limiter le nombre possible de appariements qui peuvent être évalués par l'algorithme d'objet à silhouette. ***V*** est une matrice de dimension ***n*** x ***m*** où n est le nombre d'objets suivis et m est le nombre de silhouettes. Donc ***V*** est construite par la formule :

$$V_{ji}=1 \quad \text{if} \quad \left\|\mathbf{q}_j,\mathbf{s}_i\right\|\le r \qquad (1\le j\le n, 1\le i\le m) \qquad (4)$$

où q et s sont les vecteurs représentant les position de l'objet ***j*** et la silhouette ***i*** dans l'image. Le coût de chaque application d'objet à silhouette ayant une entrée non zéro dans la matrice ***V*** est calculé par la valeur scalaire:

$$c(Q,S)=\sum_k \frac{\mathbf{d}_k(Q,S)}{\mathbf{f}_k(Q)} \qquad (5)$$

où $\boldsymbol{d}_k(\boldsymbol{Q},\boldsymbol{S})$ est le $\boldsymbol{k}^{\text{-ième}}$ élément du vecteur correspondant à la différence entre l'objet ***Q*** et la silhouette ***S***, et $\mathbf{f}_k$ est le vecteur du profil de l'objet ***Q***. Le dénominateur de la formule (5) est transformé à une valeur scalaire en calculant la longueur euclidienne de vecteur du profil. En revanche une silhouette seule peut s'apparier avec plus d'un objet, donc la matrice d'appariement ***M*** doit être initialisée par une condition selon la formule (6). Chaque affectation de l'objet à la silhouette dépend d'une entrée différente de zéro dans la matrice d'appariement valide ***V :***

$$M_{jv}=V_{jv} \quad \text{where} \quad v=\arg\min_i\left\{c(Q_j,S_i)\right\} \qquad (1\le j\le n, 1\le i\le m) \qquad (6)$$

où une entrée différente de zéro indique un appariement entre l'objet $\boldsymbol{Q}_\mathbf{j}$ et la silhouette $\boldsymbol{S}_\mathbf{i}$.

Étape 2. Suppression d'appariement duplicatif : L'appariement naïf est moins coûteux pour apparier chaque objet, mais il est possible qu'il y ait un conflit, parce qu'une silhouette est appariée initialement par plus d'un objet. Dans ce cas là il faut déterminer si l'objet est temporaire. On appelle ici les objets transitoires les objets qui sont simplement instanciés par une image. Pour être qualifié d'objet non-transitoire, l'objet doit être apparié par au moins deux images. Si on a le conflit, si la silhouette est aussi appariée avec un autre objet non-transitoire l'appariement de la silhouette à l'objet transitoire est supprimé. Cette étape sert à enlever certains faux objets qui influencent le suivi du véritable objet.

Étape 3. Évaluation de fusionnement possible : Quand de plus en plus de conflits d'appariement ont lieu sur les objets non-transitoires, l'appariement duplicatif peut être produit par la fusion des silhouettes proches. On utilise les objets affectés par une silhouette conflictuelle pour avoir un seul macro-objet $\boldsymbol{\Theta}$ avec son seul vecteur de profil, peu importe si la fusion se passe ou non. Donc si le coût d'appariement du macro-objet à silhouette est inférieur que le coût minimum d'appariement d'objet à silhouette, on peut alors supposer que la silhouette soit fusionnée. Le coût pour l'appariement de macro-objet à silhouette est calculé par la formule:

$$c(\Theta, S) = \sum_k \frac{\mathbf{d}_k(\Theta, S)}{\mathbf{f}_k(Q_j)} \quad \text{where} \quad j = \arg\min_v \left\{ c(Q_v, S_i) \right\} \tag{7}$$

Le coût de l'appariement entre macro-objet $\boldsymbol{\Theta}$ et silhouette match $\boldsymbol{S}$ est la somme des éléments dans le vecteur de différence $\boldsymbol{d}(\boldsymbol{\Theta}, \boldsymbol{S})$ scalarié par les vecteurs de profil des éléments correspondants $\mathbf{f}(\boldsymbol{Q}_\mathbf{j})$. $\boldsymbol{Q}_\mathbf{j}$ est l'objet qui apparie le mieux la silhouette, et le coût de cet appariement vaut $\mathbf{c}(\boldsymbol{Q}_\mathbf{j}, \boldsymbol{S})$ calculé par l'étape 1.

Étape 4. Suppression d'appariement duplicatif : L'appariement de l'objet non-transitoire corrigé par l'étape 3 a peut-être déjà été alloué pour la silhouette appariée par l'objet transitoire. Cela est permis car l'objet établi est prioritaire sur l'objet transitoire. On supprime l'appariement de l'objet transitoire qui entre en conflit avec l'objet non-transitoire. Chaque silhouette allouée à l'objet non-transitoire est appelée "l'appariement de sécurité", et le vecteur de différence $\mathbf{d}(\boldsymbol{Q}, \boldsymbol{S})$ est recalculé par la prochaine étape.

Étape 5. Découpage Silhouette Fusionnée : Ici la fonction de découpage de silhouette est appliquée pour résoudre une silhouette appariée à plus d'un objet non-transitoire. Si l'appariement dupliqué est passé par l'étape 3, alors il est possible que les silhouettes d'objets aient fusionnés et elles doivent se découpées pour permettre de suivre l'objet séparément.

Fonction de Découpage de Silhouette : **É**tant donnée les coordonnées (x, y) de chaque pixel de silhouette, la somme de 'ligne de régression linéaire des moindres

carrés' y = a + bx peut être calculée par la formule suivante :

$$a = \frac{\left(\sum y\right)\left(\sum x^2\right) - \left(\sum x\right)\left(\sum xy\right)}{n_p \sum x^2 - \left(\sum x\right)^2}, \quad b = \frac{n_p \sum xy - \left(\sum x\right)\left(\sum y\right)}{n_p \sum x^2 - \left(\sum x\right)^2} \tag{8}$$

Cette expression est appliquée pour tous les pixels n_p dans la silhouette.

Donc chaque pixel est projeté sur la ligne de l'expression (8), la silhouette est divisée en plaçant la ligne de découpage à l'intervalle au long de la ligne de régression, l_r. Pour calculer les points découpés, les tailles des objets participant au morceau de découpage sont listées au long de l'axe X selon leurs positions relatives. Pour n objets, il y aura *n - 1* partitions **p**. Si l'extrême gauche de la projection de silhouette sur la ligne de régression est l'origine, et que l'extrême droite est l'unité, la découpage p_m est dans l'intervalle {0, 1},

$$p_m = \frac{\sum_{j=1}^{m} a_j}{\sum_{k=1}^{n} a_k} \quad (1 \le m \le n-1) \tag{9}$$

où p_m désigne le découpage m, a_j est la surface de l'objet j et n est la nombre total des objets participant au morceau découpé.

Basé sur le label de pixel, un vecteur de profil est calculé pour chaque sous-silhouette. La différence entre les vecteurs de profil des objets participant et les nouvelles silhouettes découpées sont calculées et le coût est évalué par l'expression :

$$c(Q, \Phi) = \sum_k \frac{\mathbf{d}_k(Q, \Phi)}{\mathbf{f}_k(Q^{AV})} \tag{10}$$

où $d(Q,\Phi)$ est la différence entre objet Q et une sous-silhouette découpée Φ, le coût de l'appariement est la somme des éléments à échelle réduit de la différence vecteur.

Étape 6. Fusionner Silhouette Découpée : Un objet peut apparaître comme plusieurs silhouettes séparées. Dans l'étape 4 une silhouette appariée à un objet non-transitoire est combinée avec une silhouette sans appariement de sécurité dans le rayon spécifié. Par combiner les découpages à silhouette fusionnée **Ψ**, une silhouette découpée peut être reconstruite pour améliorer l'appariement afin de bien suivre l'objet. Donc le coût du nouveau appariement est évalué par :

$$c(Q, \Psi) = \sum_k \frac{\mathbf{d}_k(Q, \Psi)}{\mathbf{f}_k(Q)} \tag{11}$$

Si $c(Q, \mathbf{\Psi}) < c(Q, \boldsymbol{S})$, et si la silhouette fusionnée **Ψ** produit un appariement à l'objet Q plus proche que la silhouette non-modifiée **S**, le fusionnement est effectué et la silhouette est combinée dans la liste de silhouette. Cette étape va être reproduite jusqu'à ce que tous les découpages de silhouette non-allouée soient considérés.

Étape 7. Raffiner l'appariement Transitoire : Dans cette étape, seuls les objets transitoires sont examinés et ils peuvent être combinés avec les silhouettes non-allouées.

Étape 8. Mettre à jour les Objets : Étant donné une matrice d'appariement ***M***, la liste d'objets est mise à jour.

Donc cet article parle de l'algorithme de suivie l'objet basé sur différenciation entre arrière-plan et avant-plan, puis il présente la méthode pour raffiner le suivi en appariant correctement l'objet et la silhouette et pour éviter les redondances.

Exemple 1-2

Un autre article [23] « **Object Tracking with Dynamic Feature Graph** » qui présente une nouvelle représentation d'objet basée sur profil: attributed relational graph (ARG) pour suivre l'objet. Il envisage deux problèmes de suivi d'objet basé sur modèle. Le premier est de comment représenter un objet afin de bien discriminer l'objet et l'arrière-plan et d'autre objets. Le deuxième est de comment mettre à jour le modèle pour accommoder l'apparence d'objet de différente circonstance.

1.2.1 Introduction

Pour présenter l'objet, **(1)** représentation basée sur apparence, **(2)** représentation basée sur forme sont largement utilisée, et en récent **(3)** représentation de profil invariant local est développé.

La **(1)** modélise souvent l'objet grâce à histogramme couleur, la plupart sont en représentation holistique qui manque de l'information de la structure d'objet et cette représentation ne sont pas robuste pour le changement d'illumination et la confusion de l'arrière-plan.

La **(2)** modélise souvent l'objet en utilisant les 2D ou 3D modèles géométriques. Cette représentation peut représenter les structures des objets effectivement, mais elle ne peut pas bien considérer l'apparence d'objet.

La **(3)** est très réussie de reconnaître l'objet dans les patterns. Le profil local désigne qu'il est assez stable par rapport à des variations d'apparence et transformation géométrique (échelle, rotation etc.). Mais cette représentation manque de l'information sur la structure globale.

Pour surmonter les faiblesses, cet article a proposé donc un nouveau "Graphe Relationnel Attribué" (**ARG**) basé sur la représentation d'objet qui incorpore deux profils distincts – Transformation de Profil de l'Échelle Invariante (**SIFT**) et leurs relations pour suivre l'objet. Donc en local les profils décrivent le détail d'objet, puis les relations entre les profils encodent la structure d'objet en global.

Comparer avec la représentation d'objet, on a moins de travaille pour mettre à jour les modèles pour accommoder les changement d'objet. Dans l'article [5], un système de rang est proposé pour choisir le meilleur profil parmi 49 candidats. La différence entre les travaux précédents et ceux de cet article est que la méthode des travaux précédents est un suiveur discriminant qui ne peut pas facilement maîtriser l'occlusion, alors le suiveur de cet article peut le faire. Dans une séquence d'image dynamique, le profil local est peut-être instable par le temps, il peut paraître dans une image et disparaître dans une image suivante, puis il apparaît encore. On

modèle donc le comportement dynamique d'objet en utilisant un HMM d'ordre haut qui peut effectivement et d'une façon adaptative mettre à jour le modèle à traiter que le nouvel objet paraît et l'ancien objet disparaît.

Les contributions principales de cet article sont : 1) Une nouvelle présentation d'objet, compacte et robuste, basée sur profil de format **ARG**. 2) Un modèle dynamique d'objet qui peut modéliser le court terme et long terme. 3) Un MAP cadre générale est pour le suivie basée sur profil.

1.2.2 Graphe Relationnel Attribué basé sur représentation de profil

1.2.1.1 Représentation de Profil : Le profil SIFT est représenté comme : $\boldsymbol{f} = \{\boldsymbol{p, s, o, hist}\}$ où $\boldsymbol{p}$ est la position de 2-D pour le profil de coordonnée d'image, $\boldsymbol{s}$ est l'échelle de profil, $\boldsymbol{o}$ est la direction de vecteur de profil, et $\boldsymbol{hist}$ est la distribution d'orientation de gradient quantifiée à 128 pas.

1.2.2.2 Profil d'Organisation avec ARG : On définit les relations comme la suite. Supposons que $\boldsymbol{f}$ et $\boldsymbol{f'}$ soient deux profils, leurs attributs de relation est : $\boldsymbol{r(f, f')} = \{\boldsymbol{rd, rs, ro}\}$, où $\boldsymbol{rd} = |\boldsymbol{pf} - \boldsymbol{pf'}|$ est la distance euclidienne entre deux profils, $\boldsymbol{rs} = |\boldsymbol{sf} - \boldsymbol{sf'}| / |\boldsymbol{sf} + \boldsymbol{sf'}|$ est la différence d'échelle et $\boldsymbol{ro} = |\boldsymbol{Of} - \boldsymbol{Of'}|$ est la différence d'orientation. Le graphe relationnel attribué G est défini comme:

$\boldsymbol{G} = \{\Sigma \boldsymbol{f}, \Sigma \boldsymbol{r}\}$ est le graphe relationnel.

$\Sigma \boldsymbol{f} = \{\boldsymbol{f1}, \boldsymbol{f2}, ..., \boldsymbol{fn}\}$ est l'ensemble de sommets. Les profils définis dans 2.1

$\Sigma \boldsymbol{r} = \{\boldsymbol{r1}, \boldsymbol{r2...rm}\}$ est l'ensemble d'arête.

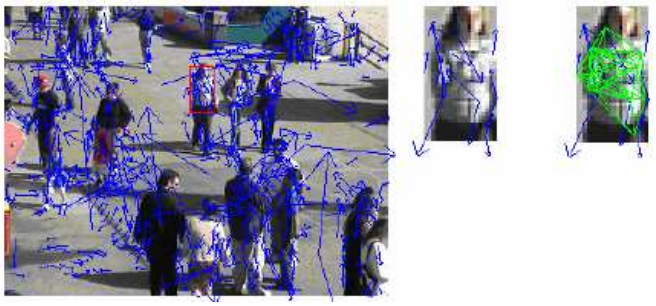

Figure 1-9 *Représentation d'objets*

Dans la figure 1-9, l'image à gauche est une scène avec le SIFT profils imposé par le cadre bleu. L'image au milieu est une version d'échelle de la zone encadrée par le rec-tangle rouge. Et l'image à droite est le graphe relationnel, avec les arêtes présentées par les lignes vertes.

1.2.3 Le Cadre MAP

Le suiveur basé sur profil est formulé à un cadre de Maximum A Posteriori (**MAP**) en utilisant Modèle de Markov Caché. Donc l'état d'objet au temps t est identifié comme :

$$X_t = G_t = \sum_f$$

On peut ignorer les arêtes (relations), car elles sont complètement dépendantes des sommets, donc elles sont déterminées dès que les sommets sont déterminés.
Particulièrement le suiveur de profil est formulé pour trouver la probabilité de MAP :

$$\max_{X_t} \arg P(X_t \mid I_t, ...I_0, X_{t-1}, ..., X_0)$$

Supposons un Modèle de Markov de l'ordre m :

$$\max_{X_t} \arg P(X_t \mid I_t, ...I_0, X_{t-1}, ..., X_0)$$
$$= \max_{X_t} \arg P(X_t \mid I_t, I_{t-1}, X_{t-1}, ..., X_{t-m})$$
$$= \max_{X_t} \arg P(I \mid X_t, I_{t-1}, X_{t-1}) P(X_t \mid I_{t-1}, X_{t-1}, ..., X_{t-m})$$

Où $\boldsymbol{P(I \mid X_t, I_{t-1}, X_{t-1})}$ est la probabilité et $\boldsymbol{P(X_t \mid I_{t-1}, X_{t-1}, \ldots, X_{t-m})}$ représente les dynamiques d'objet .

1.2.4 Dynamique d'objet et probabilité

Dans cette représentation basée sur profil, l'observation de chaque cadre est le profil de SIFT extrait, qui est utilisé pour générer le graphe relationnel. La dynamique d'objet exprime comment l'objet évolue. On modèle le graphe de dynamique à court terme et aussi à long terme. La probabilité ici est pour mesurer comment l'objet s'adapte à l'observation.

1.2.4.1 Dynamique d'Objet : Comme les sommets et les arêtes de graphe évoluent en temps, donc il faut les modéliser simultanément. En revanche les arêtes sont déterminées dès que les sommets sont déterminés. Donc on n'a besoin que de modéliser la dynamique de sommet. Donc on formule la dynamique d'objet :

$$P(X_t \mid I_{t-1}, X_{t-1}, ..., X_{t-m}) = P(\sum_f^t \mid \sum_f^{t-1}, ..., \sum_f^{t-m})$$

Supposons $\sum_f^t = \{f_1^t, f_2^t, ..., f_{Nt}^t\}$,

l'état de transition peut être factorisé selon la formule suivante :

$$P(\sum_f^t \mid \sum_f^{t-1}, ..., \sum_f^{t-m}) = \prod_{i=1}^{N} P(f_i^t \mid f_i^{t-1}, ..., \sum_f^{t-m})$$

$$P(f_i^t \mid f_i^{t-1}, ..., f_i^{t-m})) = \begin{cases} P_s(f_i^t \mid f_i^{t-1}) & when\ f_i^t \neq \varnothing, f_i^{t-1} \neq \varnothing \\ P_{new}(f_i^t \mid f_i^{t-1}, ..., f_i^{t-m}) & when\ f_i^{t-1} = \varnothing \\ P_{delete}(f_i^t \mid f_i^{t-1}, ..., f_i^{t-m}) & when\ f_i^t = \varnothing \end{cases}$$

$\boldsymbol{P}_s$ est la dynamique de court terme qui modélise la priorité d'attribut de profil. Pour mieux modéliser la dynamique de graphe, on introduit les ***profils stables***. Ils sont les profils qui sont persistants. Les profils stables peuvent modéliser les comportements d'objet. $\boldsymbol{P}_{new}$ et $\boldsymbol{P}_{delete}$ modélisent la naissance et la mort des profils stables. Ils dépendent de la performance d'appariement des profils dans les images précédentes, parce que l'on ne veut que les profils stables qui restent dans l'ensemble de profils stables.

1.2.4.1.1 Priorité d'attribut de profil: On considère que les priorités de dynamique pour la position de profil, l'échelle de profil et l'orientation de profile sont

gaussiens avec l'attribut d'état antérieur comme la moyenne, la priorité de l'attribut peut être décrite par la formule suivante :

$$P_s(f_i^t \mid f_i^{t-1}) = P(\mathbf{p}_t \mid \mathbf{p}_{t-1}) * P(s_t \mid s_{t-1}) * P(o_t \mid o_{t-1}) * P(\mathbf{h}_t \mid \mathbf{h}_{t-1})$$

$$\propto \exp\{-(\mathbf{p}_{t-1}-\mathbf{p}_t)^T \textstyle\sum_p^{-1} (\mathbf{p}_{t-1}-\mathbf{p}_t)\} * \exp\{-(\frac{s_t - s_{t-1}}{\sigma_s})^2)\} * \exp\{-(\frac{o_t - o_{t-1}}{\sigma_o})^2)\} *$$

$$\exp\{-(\frac{d(\mathbf{h}_t, \mathbf{h}_{t-1})}{\sigma_h})^2)\}$$

où p, s, o, h sont la position de profil, d'échelle, d'orientation et d'histogramme.

1.2.4.1.2 Ajouter de nouveaux profils stables: Dans chaque image, de nouveaux profils peuvent apparaître à cause de changements d'apparence ou de pose. Il n'est pas nécessaire de traiter les nouveaux profils de même manière que pour les profils stables prouvés. Donc ce qu'on va faire, c'est de garder temporairement les nouveaux profils, et après un certain moment de compétition, on ajoute les nouveaux profils vraiment stables dans le modèle. De ce fait on maintient l'ensemble de candidats de profil potentiellement stable. Chaque profil a un vecteur de statut associé pour identifier sur quelle image il apparaît. On suppose que la probabilité d'être ajoutés dans le modèle pour les profils dans l'ensemble de candidats soit la distribution binomiale, soit B(*m*, *pb*) où m est l'ordre de HMM, par exemple, le temps passé pour évaluer la stabilité de profil. Le *pb* est la probabilité d'être observé pour le profil.

1.2.4.1.3 Suppression de profil inactif: A cause de changement d'illumination ou de pose d'objet, les profils qui sont stables dans les images précédentes vont peut-être devenir inactifs dans les images suivantes. Donc l'idée de supprimer les profils inactifs est introduite dans notre cadre. Pour les profils déjà dans le modèle, on maintient quand même une historique pour leur performance dans les ***m*** images précédentes. Si un profil n'a pas réussi d'être apparié pour certaines fois, on considère qu'il est inactif et on le supprime du modèle. Même procédé pour le scénario d'incorporation de nouveaux profils. On modélise aussi la suppression du profil comme la distribution binomiale : $\boldsymbol{P}_{\text{delete}} \sim B(\boldsymbol{n}_m : \boldsymbol{m}, \boldsymbol{p}_m)$ tronqué entre $[\tau_d, 1]$, τ_d est définit comme la probabilité minimum que le profil soit supprimé du modèle.

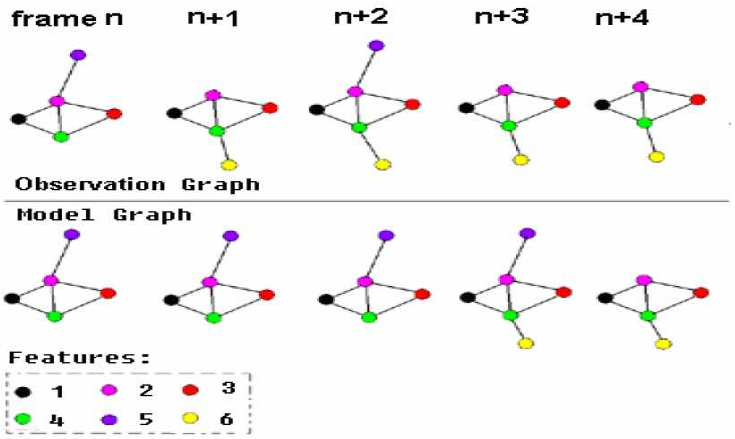

Figure 1-10 *Mis à jour pour le modèle dynamique*

La Figure ci-dessus présente les sommets colorés et les arêtes entre eux. La partie du

haut est l'observation de la séquence d'images, et la partie du bas montre les modèles d'objet sur chaque image. A l'image n (*frame n sur la Figure*), le modèle contient 5 profils (1 – 5 sommet). Dans les 5 images(n ~ n+4), le profil bleu (5) est observé seulement 2 fois, donc il est choisi comme un profil instable et supprimé du modèle (*partie en bas*) dans l'image n+4. Dans l'image n+3, le nouveau profil 6 a été toujours observé pour 3 images consécutives, donc il est un profil stable et sélectionné à partir de l'ensemble de candidat.

1.2.4.2 Probabilité : La fonction de probabilité $\boldsymbol{P}(\boldsymbol{I}_t \mid \boldsymbol{X}_t)$ décrit comment l'état $\boldsymbol{X}_t$ s'adapte à l'observation $\boldsymbol{I}_t$. La fonction de probabilité basée sur le profil est calculée pour une note de l'appariement entre la représentation d'objet et l'observation.

1.2.4.2.1 Formulation d'appariement de graphe : Pour maîtriser le cas où il y a différent nombre de profils dans deux graphes, on ajoute des sommets factices dans l'ensemble de sommets du graphe d'observation $\sum_f$ et l'ensemble de sommets du graphe de modèle $\sum_f^M$. Donc pour chaque fonction d'appariement F, on permet d'apparier le profil avec le sommet factice quand aucun bon appariement ne peut être réalisé. Alors la probabilité peut être mesurée comme la similarité entre l'observation et le graphe de modèle. On suppose que le modèle d'observation soit :

$$f_{Gi} = f_{Mi} + e(i)\ ;\ r_{Gi} = r_{Mi} + e'(i)$$

où $\boldsymbol{f_{Gi}}$ et $\boldsymbol{f_{Mi}}$ sont les profils du graphe d'observation et du graphe de modèle, $\boldsymbol{r_{Gi}}$ et $\boldsymbol{r_{Mi}}$ sont les relations du graphe d'observation et du graphe de modèle, *e(i)* et *e'(i)* sont des bruits gaussiens indépendants de moyenne zéro (zero mean). La probabilité peut être formalisée comme une distribution de Gibbs avec énergie :

$$E(G \mid X) = E(G \mid F) = \sum_{f \in \sum_{Gf}} E_1(f \mid F(f)) + \alpha \sum_{r \in \sum_{Gr}} E_2(r \mid F(r))$$

où $\boldsymbol{P_1}$ et $\boldsymbol{P_2}$ sont les potentiels de profil et d'arête, et α est le coefficient pour balancer l'appariement de profil et l'appariement de relation :

$$E_1(f_i \mid F(f_i)) = \begin{cases} \sum_{i=1}^{K1} [f_i^k - F(f_i)^k]^2 / [\sigma_f^k]^2 & \text{if } F(f_i)^k \text{ is not null} \\ P_{v1} & \text{if } F(f_i)^k \text{ null} \end{cases}$$

où $\boldsymbol{F}(f_i)$ est le sommet correspondant à f_i, et K1 est le nombre d'attributs relatif au sommet f_i . Le f_i^k est le k-ième élément des attributs. Et $(\sigma_i^k)^2$ est la variance de sa distribution de bruit gaussien. Si l'appariement de profil est un sommet factice, on affecte une large sanction $\boldsymbol{P}_{v1}$.

$$E_1(r_i \mid F(r_i)) = \begin{cases} \sum_{i=1}^{K2} [r_i^k - F(r_i)^k]^2 / [\sigma_r^k]^2 & \text{if } F(r_i)^k \text{ is not null} \\ P_2(r_i \mid F(r_i)) = P_{v2} & \text{if } F(r_i)^k \text{ is null} \end{cases}$$

où $\boldsymbol{F}(\boldsymbol{r_i})$ est l'arête correspondante de r_i , et K_2 est le nombre de proprités relatif à

l'arête e_i, e_i^k est le k-ième élément des propriétés, et $(\sigma_i^k)^2$ est la variance de la distribution de bruit gaussien. Si l'arête est appariée à une arête factice, une sanction P_{v2} lui est affectée.

1.2.4.2.2 Étiquetage de relaxation pour l'appariement de graphe: En général l'appariement de graphe est un problème de NP-Complet. En premier lieu, on utilise la distance de profil pour initialiser l'appariement de graphe (sans con-sidérer les relations), puis on rafine l'appariement par une étiquétage de relaxation.

On construit d'abord un graphe plus grand que celui dans l'état précédent, puis on utilise l'appariement de sous-graphe pour apparier le modèle avec ce graphe, illustré dans la figure 1-11. Pour cela on utlise la méthode d'étiquetage de relaxation. L'idée est d'utiliser la mis à jour de context local répétée pour avoir globalement un résultat constant.

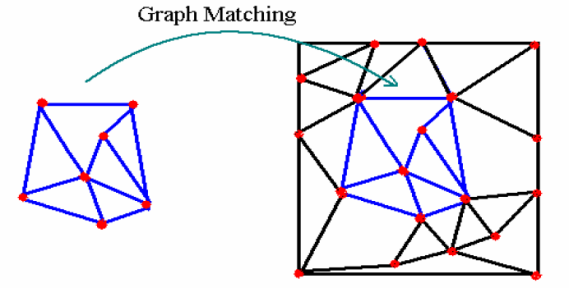

***Figure 1-11**. L'appariement de graphe*
A gauche c'est le graphe de modèle.
A droite c'est le graphe construit plus grand que l'état précédent.

La méthode d'étiquetage de relaxation considère la compatibilité des probabilités d'étiqueter comme les contraints dans l'algorithme d'étiquetage, i.e., prendre en compte ses voisins. La compatibilité $C_{ij}(f'_k, f'_l)$ est définie comme la probabilité conditionnelle où le profil f_i ayant une étiquette f'_k donne le profil f_j ayant l'étiquette f'_l, i.e. $C_{ij}(f'_k, f'_l) = P(f'_k \mid f'_l)$. La mis à jour des probabilités d'étiquettes est faite en considérant les probabilités d'étiquettes pour les profils de voisin. Supposons que l'on ait changé toutes les probabilités jusqu'à certaine étape, on cherche une probabilité de mis à jour pour la prochaine étape. Alors on peut estimer le changement de $P_i(f'_k)$ par :

$$\delta P_i(f'_k) = \sum_{j \in N} w_{ij} \left[\sum_{l \in L} C_{ij}(f'_k, f'_l) P_j(f'_l) \right]$$

où N est le nombre de voisins de f_i, et wij est le poids de ces voisins avec la contrainte Σ wij = 1. La nouvelle probabilité pour étiqueter $P_i(f'_k)$ dans la génération de prochaine étape peut être calculée à partir des valeurs utilisées dans l'itération précédente :

$$p_i(f'_k) = \frac{p_i(f'_k)[1 + \delta p_i(f'_k)]}{\sum_l p_i(f'_l)[1 + \delta p_i(f'_l)]}$$

De cette façon, après certaines itérations la probabilité de l'étiquetage de profil va se stabiliser, donc l'appariement de profil à profil est obtenu.

Chapitre 2

La Stabilisation d'Image

Dans ce chapitre on va s'intéresser aux études de stabilisation d'image. Dans l'orientation de recherche de ce mémoire, la stabilisation d'image peut aussi servir à localiser la caméra.

Généralité:

La stabilisation d'image est un ensemble de techniques pour produire une image stable à partir d'une optique non stable. Elle est largement utilisée dans les domaines comme : image stabilisée binoculaires, photographie, vidéographie, astronomie, etc. Avec une caméra photographique, le secouement de camera est un problème particulier en cas de vitesse lente de déclencheur ou bien à cause de la lentille de longue longueur focale de l'objectif, par exemple le téléobjectif pour téléphotographie. Avec la caméra de vidéo, le secouement de caméra cause le tremblement visible d'image à image dans l'enregistrement de vidéo. En astronomie, ces problèmes sont combinés par les variations dans l'atmosphère au fils du temps, qui provoque le changement de la position apparente d'objet.

Les principales techniques pour la stabilisation d'image sont :

1) Stabilisation optique d'image, 2) Anti-Secouement,
3) Stabilisation numérique d'image 4) CCD de transfert orthogonal.

La première est un mécanisme utilisé dans la caméra photo numérique ou la caméra de vidéo. Il stabilise les images par la variation du chemin optique pour le senseur. Cela fonctionne par le biais d'un élément de lentille flottante qui est actionné en relation avec la lentille stable, le gyroscope est souvent utilisé dans ce cas-là.

La deuxième est une technique utilisée par Konica et Sony. Le senseur de CCD (charge-coupled device) peut être bougé pour contrarier le mouvement de caméra. Ce système repose sur la mis en œuvre d'un senseur très précis de la vitesse d'angle pour détecter le mouvement de caméra.

La troisième est utilisée dans certaines caméras. Cette technique transforme de l'image en l'image les images électroniques sur la vidéo, suffisamment pour contrarier le mouvement apparent. Elle utilise les pixels à l'extérieur du bord d'image

visible pour apporter un tampon aux mouvements.

La quatrième est utilisée en astronomie. En effet un CCD de transfert orthogonal transforme les images au sein de CCD lorsque l'image est en train d'être capturée. Cette technique est basée sur l'analyse du mouvement apparent de l'étoile brillante. Un exemple de cette technique est Pan-STARRS utilisé pour le télescope de gigapixel, il est en train d'être construit sur les îles Hawaii.

Pour notre recherche:

D'après les principales techniques de stabilisation d'image principales, on voit bien que ce domaine consiste à traiter les images en mouvement. Ce mouvement d'image peut être causé par l'objet en mouvement ou/et par la caméra en mouvement. A cause de ces mouvements les images sont éventuellement floues. Pour notre recherche actuelle, les méthodes de la troisième technique sont plus intéressantes. A partir des images floues capturées directement par la caméra, on va procéder à des compensations pour contrarier le mouvement apparent dans ces images, afin que les images ne soient plus floues.

Revenons sur notre sujet. Dans le chapitre précédent on utilise le " suivi d'objet " pour détecter le mouvement de la caméra. Et ici, pour stabiliser l'image numérique, on doit identifier le mouvement de l'objet ou de la caméra qui rend les images floues. Grâce à la technique de stabilisation d'image, nous pouvons détecter le mouvement de l'objet dans le champ de la caméra et/ou le mouvement de la caméra.

Exemple 2-1

L'article [6] « Image Stabilization by Features Tracking » nous apporte une technique pour stabiliser les images dans une séquence de vidéo. Le gondolage (warping) comme une compensation pour le mouvement de caméra est calculé par le profil suivi dans les images. Afin de traiter les objets en mouvement, il existe une technique fiable pour calculer l'homographie. D'après [7], [8], [9], on suit un ensemble de profils d'une séquence, et utilise les mouvements d'image pour estimer le gondolage stabilisant. D'ailleurs le [10] utilise directement l'intensité d'image dans l'approche de grossièreté vers finesse, pour le suivi d'une région unique. Le [11] utilise un filtre pour prédire la position du profil. Et dans cet article [6], on le modifie, et on adopte une règle de rejet rapide des étrangers (outlier) afin d'estimer l'homographie d'une manière fiable. De cette manière un objet en mouvement dans un arrière-plan statique peut être traité.

Le suivi d'objet a aussi un avantage de pouvoir calculer le gondolage global en chaque image, il peut prédire la position des profils perdus. La stabilisation d'image est une technique très proche de la mosaïque, la séquence stabilisée produit une mosaïque par un fusionnement direct des images. Quelques auteurs [12], [13], [14] et [15] utilisent l'appariement de profil pour construire les mosaïques. Dans l'article [6], est décrite une technique similaire utilisée dans le cadre de stabilisation d'image.

2.1.1 Compensation de mouvements

Si la caméra capture une scène approximativement plane (par exemple une vue aérienne), les points de l'image correspondante sont liés par une transformation projective linéaire, c'est l'homographie. Afin de compenser le mouvement relatif de la caméra, on a besoin de calculer les homographies qui tracent chaque image sur une image de référence donnée. On suppose que chaque image de la séquence chevauche celle de référence.

Supposons que les correspondances de point par la séquence d'image sont données, la matrice d'homographie **M** qui lie deux points correspondants pi et pj dans deux images génériques f_i et f_j est donc définie par formulation :

$$\mathbf{p}_i = \mathbf{M}\mathbf{p}_j$$

ou bien :

$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ w_j \end{bmatrix}$$

où les points sont représentés par les coordonnées homogènes, cela veut dire que l'on indique les points de 2-D dans le plan d'image comme (x, y, w) avec (x/w , y/w), ils sont les coordonnées cartésiennes correspondantes. Chaque point correspondant génère deux équations, puis les n ≥ 4 points génèrent 2n équations linéaires qui est

suffit à résoudre **M**. Le système surcontraint est facilement résolu par calcul de La matrice pseudo-inverse de système.

Pour rendre la procédure de stabilisation moins sensible aux échecs potentiels des suiveurs ou aux profils attachés aux objets en mouvement indépendant, on emploie une règle de rejet fiable (X84) pour identifier les 'étrangers'. Cela concerne des profils qui ont leur mouvement en désaccord avec celui dominant (le mouvement plan des profils majoritaires). Si $\hat{M}$ est l'homographie carrée moindre, le résidu de ***i***ième profil est défini par:

$$r_i = ||\mathbf{p}_i - \hat{\mathbf{M}}\mathbf{p}_j||$$

2.1.2 Suivi de profils

Le suivi de profils fait l'appariement en choisissant les profils d'image et suit ces derniers quand ils se déplacent d'une image à une autre. Cela peut être considéré comme un problème de calcul du flot optique. La méthode est basée sur les profils de deux dimensions, elle a l'avantage que le mouvement d'image calculé n'est pas perverti par le problème d'ouverture.

Les profils sont extraits dans la première image, et puis ils sont suivis dans chaque image d'une même sous-séquence en utilisant un filtre linéaire de Kalman [16] pour estimer et prédire leur trajectoire. Supposons que le mouvement de profil est presque linéaire dans une tranche de temps, en effet l'expérimentation a prouvé cette hypothèse.

On suppose que les images f_0, f_1, f_2, ... , f_k, ... sont capturées par une caméra avec une vitesse $1/\Delta t$. Le vecteur d'état de Kalman linéaire est défini comme:

$$\mathbf{x}_k = [\ x_k \quad y_k \quad u_k \quad v_k\ ]^T$$

où (x_k, y_k), et (u_k, v_k) sont respectivement la position et la vélocité de chaque point de profil dans l'image ***k***. La matrice de transition d'état et la matrice de mesure sont représentées respectivement:

$$\Phi = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

et

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Les matrices de covariance **Q** et **R** modèlent respectivement l'incertitude pervertissant la prédiction et l'incertitude pervertissant la mesure. **Q** dépend de l'image dérivative, et **R** dépend de la valeur de corrélation entre les fenêtres des profils courantes et celles mises à jour.

Le vecteur d'état est associé à la matrice de covariance d'état $\mathbf{P}_k$, elle encode l'incertitude de l'état courant. La région de l'espace de phase au tour de l'état

estimé $\hat{x}$ qui contient le vrai état avec une probabilité c^2, est donnée par l'ellipsoïde :

$$(\mathbf{x}-\hat{\mathbf{x}})(\mathbf{P}_k)^{-1}(\mathbf{x}-\hat{\mathbf{x}})^T \leq c^2$$

Afin de trouver la position du profil donné dans l'image précédente, on prend une petite fenêtre au tour de ce profil et on cherche le minimum de l'erreur de SSD (Somme de Différence Carrée) près de la position prédite. S'il arrive un échec de l'appariement, une recherche est effectuée aux alentour de cette position dans l'image précédente.

Si l'appariement n'a toujours pas réussi, ce profil non apparié est appelé "ghost", et il va être conservé virtuellement pendant certaines images de sous-séquence. Soit il apparaît plus tard, sinon on l'abandonne. En général la durée de vie du "ghost" doit être bien choisi, et être raisonnablement courte.

2.1.3 Construction de mosaïque

La compensation efficace de mouvement est démontrée par la construction de l'image en mosaïque. La mosaïque est une image obtenue en alignant et fusionnant d'autres images qui montrent une portion différente de la même scène.

Avec une séquence de n images, une image de mosaïque est construite en plaçant l'image de référence au centre de la mosaïque et puis en y ajoutant chaque nouvelle image stabilisée. Dans le cadre de mosaïque, cette technique est appelée la méthode "de l'image à la mosaïque". D'ailleurs il existe aussi la méthode "de l'image à l'image" et la méthode "de la mosaïque à l'image".

L'auteur de l'article [6] considère que l'utilisation de la dernière image semble être la technique la plus appropriée, mais à cause de quelque désalignements, distorsions radiales, et changements d'illumination, il pourrait y avoir une discontinuité de la correspondance de la bordure de l'image en mosaïque. Enfin on utilise la moyenne de poids comme une fonction de mélange, tel que le poids d'un pixel dans l'image à être mélangée diminue avec la distance du centre.

Exemple 2-2

Ensuite l'article [17] « Real-Time MMX-Accelerated Image Stabilization System » va être présenté. Cet article parle d'une démonstration d'un système de stabilisation d'image (SSI) pour enlever ou diminuer des mouvements apparents dans la vue indésirables en temps réel. Ces mouvements apparents sont produits à cause de mouvement de caméra, par exemple des secouements de la main tenant la caméra ou des vibrations de caméra sur un véhicule bougeant. Des applications de ce système (SSI) essaient d'enlever les mouvements apparents.

Au niveau d'implémentation de ce système (SSI), on utilise le langage C++ pour coder, et puis MMX-accélération pour achever la performance en temps réel avec des matériaux moins coûteux. MMX est l'Extension Multimédia.

2.2.1 Algorithme de Stabilisation d'Image

2.2.1.1 Modèle:

On suppose que les images soient capturées par une caméra qui est portée par un transporteur mobile comme un véhicule ou une personne marchant. On peut donc apparier les images successives pour trouver le mouvement de la caméra entre les images. Ce mouvement est mélangé par la vibration et le mouvement principal de caméra. Il vaut mieux de séparer la vibration du mouvement principal. La vibration a souvent un spectre de haute fréquence, il est plus haut que la fréquence de spectre du mouvement global de caméra. Donc on peut utiliser le filtrage de mouvement de basse fréquence [18] pour enlever ce bruit de haute fréquence.

Après l'enlèvement du bruit de la vibration, le mouvement de caméra est considéré comme un mélange de la translation et la rotation, voir la figure 2 – 1.

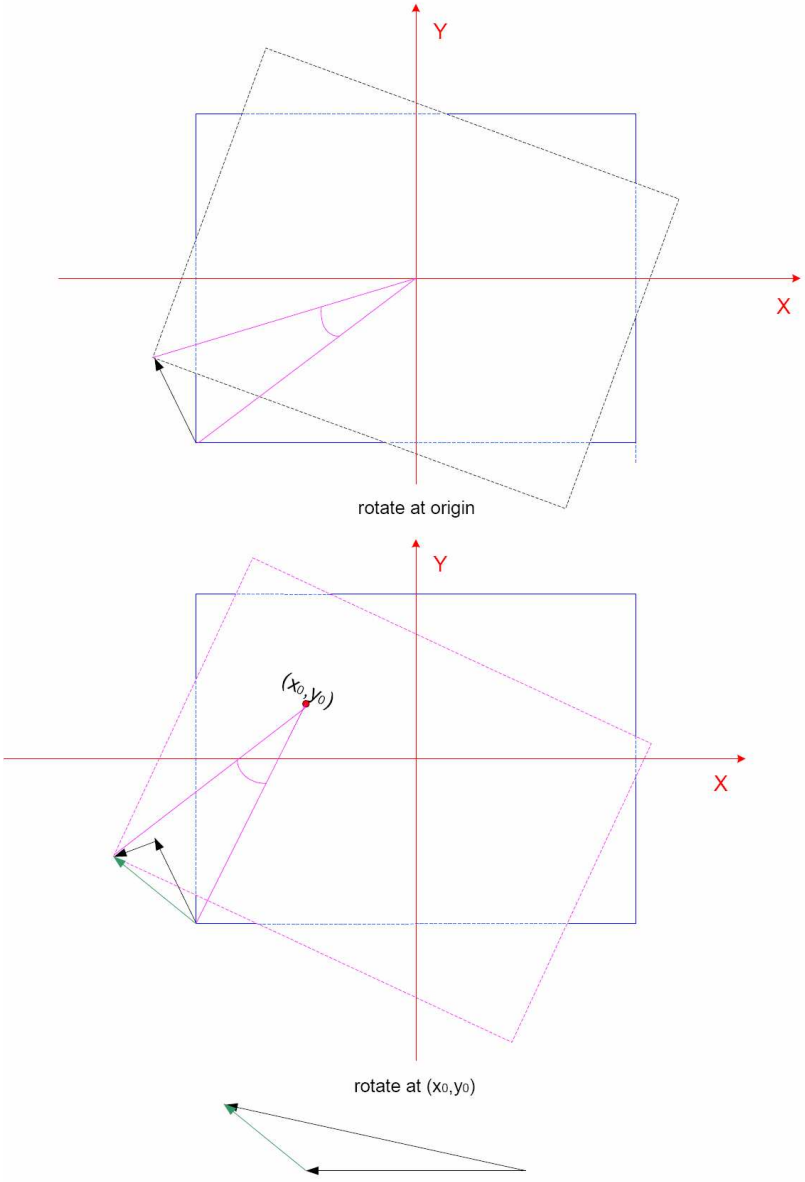

Figure 2-1 *Rotation entre deux images adjacentes*

Chaque mouvement est représenté par le vecteur de translation **γ** et l'angle de rotation **θ**. Si le centre de rotation est à x_0, y_0 , donc le mouvement total Ω est donc

$$\Omega = \Upsilon + \nu + (y_0 \sin\theta - x_0 \cos\theta, -x_0 \sin\theta - y_0 \cos\theta)$$

où v = (***x***cos **θ** – ***y***sin**θ** , ***y***cos**θ** + ***x***sin **θ**), et (***x***, ***y***) est la coordonnée originale de chaque point dans l'image.

Ainsi une rotation avec le centre (x_0y_0) peut être considérée comme une rotation à l'origine avec le même angle plus un vecteur de translation, soit :
δ = ($\boldsymbol{y_0}$sin **θ** – $\boldsymbol{x_0}$cos**θ**, – $\boldsymbol{x_0}$sin **θ** – $\boldsymbol{y_0}$cos**θ**). Afin d'estimer le mouvement **γ** et l'angle **θ** de rotation de l'image, on a besoin d'obtenir l'information suffisant de mouvement à partir des différents points dans les deux images successives. Chaque point ***P*i** a un vecteur de mouvement **ω*i*** = **v** *i* + ***v*** *i* + δ. Si ces points sont symétriques, puis:

$$\begin{aligned}\sum_{i=1}^{N} \omega_i &= \sum_{i=1}^{N} \upsilon_i + \sum_{i=1}^{N} \nu_i + N * \delta \\ &= \sum_{i=1}^{N} \upsilon_i + 0 + N * \delta.\end{aligned}$$

Le vecteur de moyen mouvement $\bar{\omega}$ est donc:

$$\begin{aligned}\bar{\omega} &= \frac{1}{N}(\sum_{i=1}^{N} \omega_i) \\ &= \frac{1}{N}\sum_{i=1}^{N} \upsilon_i + \delta.\end{aligned}$$

On a l'angle de rotation **θ** de chaque vecteur **ω*i***.

2.2.1.2 Problème d'Implémentation :

La méthode la plus utilisée pour déterminer le mouvement entre deux images est l'appariement de bloc. Mais si on apparie tous les blocs de l'image comme le [18], cela va mettre très long temps pour traiter toutes les images et cela n'est plus stabilisation en temps réel. Afin de le faire assez rapidement pour que cela soit en temps réel, on peut utiliser une étape de détection du bord [19][20]. Elle peut abstraire les profils à partir des images pour faire l'appariement plus précis au cas où l'appariement de bloc échoue à cause du contraste bas.

Concernant la méthode de l'appariement on trouve que c'est mieux d'utiliser l'appariement différentiel minimum, car il est très rapide et assez précis, surtout pour des régions contenant le bord.

Enfin il faut bien choisir la taille de bloc. La taille doit être la plus petite que possible pour que l'appariement de bloc soit efficace. Mais quand elle est trop petite, le bloc peut-être ne contient pas suffisamment de profils et cela peut causer l'appariement imprécis, surtout cela sera grave avec la rotation d'image. Finalement on a choisi la taille supérieure de 25 x 25.

2.2.2 Structure du Système de Stabilisation d'image

Étape 1) Détection du Bord : Dans cette étape on convertit d'abord l'image couleur en niveaux de gris. Puis on utilise le filtre Sobel horizontal et vertical pour détecter le bord. L'image est divisée en 16 parties, et on peut alors déterminer les blocs avec le plus de profils pour savoir le mouvement, on met donc un seuil. Si aucun bloc ne contient assez de bords dans certaine zone, on ignore cette zone. Pour optimiser l'appariement de l'étape suivante, on essaie de prendre les blocs qui sont symétriques entre eux par l'origine.

Étape 2) Détection du Mouvement : Cette étape est critique pour ce système. On utilise l'appariement de bloc pour détecter le vecteur de mouvement de tous les blocs valides autours du points de référence. Puis on peut enlever les vecteurs de blocs qui sont évidement largement loin des autres. On met aussi un seuil pour supprimer le mouvement et la rotation estimée trop large. Ici on met 5°.

Étape 3) Compensation du mouvement : Après l'extrait de tous les vecteurs de mouvement, on filtre la vibration et la rotation de ces vecteurs. On calcule une moyenne des vecteurs de mouvement p_i, pour avoir le vecteur de mouvement global $\bar{p}$ de cette image. Il est combiné par le mouvement de transporteur actuel et le secouement. Et on utilise le filtrage de mouvement de basse fréquence [18] sur $\bar{p}$ pour enlever le bruit de haut fréquence et avoir le mouvement estimé $\boldsymbol{c}$.
Puis le vecteur de mouvement de vibration $\boldsymbol{s} = \bar{p} - \boldsymbol{c}$. Et on fait une prédiction dans ce système (SSI), donc le système va donc estimer la position de l'image suivante et établir un point de référence pour apparier selon cette prédiction afin d'accélérer le traitement. Pour l'angle de la rotation on a le vecteur de l rotation: $\sin\theta_i = \frac{r_i + \delta}{L_i}$, où L_i est la distance entre le centre de i-ième bloc et l'origine. Pour obtenir δ on utilise quelques paries de vecteur des blocs symétriques,

$$\frac{r_m + \delta}{L_m} = \frac{r_n + \delta}{L_n}$$

$$\delta = \frac{r_m L_n - r_n L_m}{L_m - L_n}$$

La moyenne de l'angle θ est $\bar{\theta} = \frac{1}{N}\sum_{i=1}^{N}\theta_i$. Après on intègre le vecteur s et l'angle de la rotation $\bar{\theta}$ pour la compensation des images.

Chapitre 3

La Souris Optique

Comme nous avons déjà évoqué dans l'Introduction, on peut avoir la possibilité de créer une SourisPhone grâce à un téléphone portable qui pilote une caméra activée. Nous allons donc étudier maintenant la souris optique pour voir si son mécanisme de détection du mouvement peut servir à cette recherche.

3.1 Généralité

La souris optique est basée sur un senseur optique composé par 4 parties principales: matériel d'illumination, lentille d'illumination, caméra, lentille de la caméra. Voir la figure 3-1:

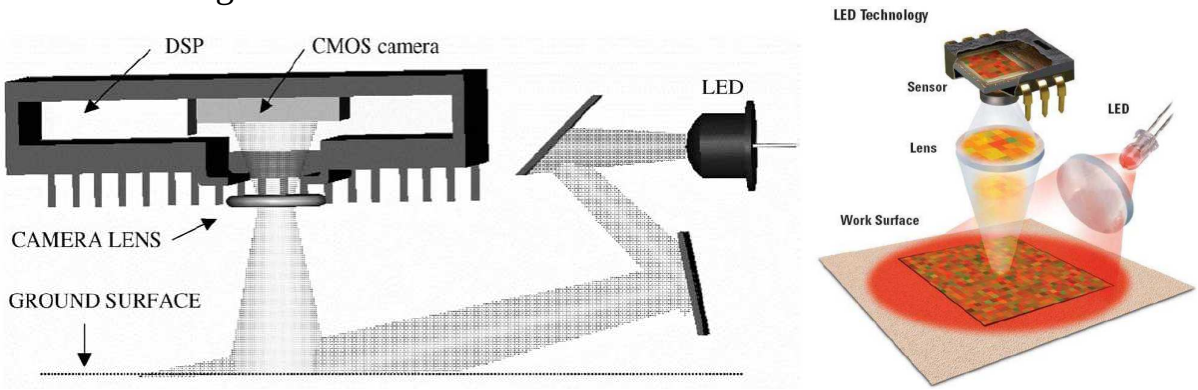

Figure 3-1 *Les composants de la souris optique*

Une LED (diode émettrice de lumière) avec une longueur d'onde de 639 à 875 nms combinée par la lentille plastique et les miroirs est utilisée pour illuminer la surface au-dessous du senseur. Il y a une caméra de CMOS (semi-conducteur de métal dioxyde complémentaire) pour acquérir les images, et il y a aussi un DSP (processeur de signal numérique) prioritaire pour le traitement des images et la communication externe. En plus une lentille plastique collecte la lumière reflétée et forme l'image dans la caméra. La lentille d'illumination qui fournit l'illumination latérale de la surface dans la Figure 3-1, utilise un filtre de la fréquence spatiale pour avoir une grande maîtrise dans la mesure.

La lumière de la LED reflète le profil textural microscopique de la zone, et la caméra prend des instantanés de la surface. Selon la catégorie de senseur, les images acquises peuvent être entre 16x16 et 24x24 pixels carrés. L'acquisition peut être répétée avec une vitesse de 1500 à 2500 images par second.

Le DSP tourne un algorithme breveté (US6281882) de traitement d'image. Cet algorithme identifie la texture ou d'autres caractéristiques entre les images et calcule la distance et le mouvement entre ces images.

3.2 Mécanisme du calcul de mouvement

Comment le DSP traite les images pour calculer le mouvement ? Voilà une question qui nous intéresse dans ce mémoire. Au début du développement de la souris optique, on avait besoin d'un tapis express pour que la souris acquière des images spéciales. Ces images apparaissent comme des plans de grille, voir la figure 3-2.

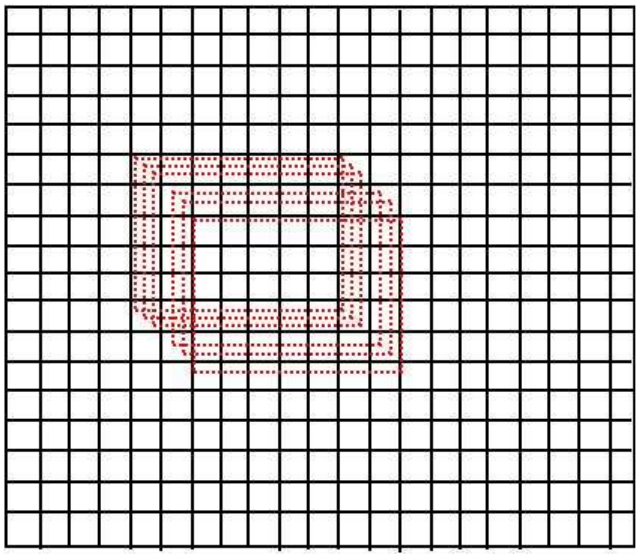

Figure 3-2 *la grille agrandie du tapis de la souris optique*

Les lignes sur la grille sont assez denses et la distance entre deux lignes adjacentes est assez petite pour avoir une grande sensibilité de détection du mouvement. Sur les images, il y a 4 côtés, deux verticales et deux horizontales. La souris optique acquiert les images avec une haute fréquence pour assurer que deux images successives ont une même ligne de côté horizontal et une même ligne de côté vertical. Ainsi le DSP calcule la distance de la ligne commune horizontale et verticale entre deux images pour avoir le vecteur de mouvement.

3.3 Pour notre recherche:

Cette méthode est aussi utilisable pour notre objectif. Dans le chapitre du "Suivi d'Objet", on a introduit la feuille à carreaux, mais ici on utilise une petite feuille de grille pour faire une SourisPhone en mouvement sur une petite zone. On peut également utiliser une grande feuille de grille qui permet une très grande zone de mouvement, comme un plan ayant plusieurs particules dans le chapitre du "Suivi d'Objet".

Chapitre 4

Le Flot Optique

4.1 Introduction

Le terme du flot optique a été inventé par le psychologue James GIBSON dans une étude sur la vision humaine. La mesure du flot optique est une étape de traitement de l'image dite de bas niveau. Le flot optique peut être utilisé pour calculer la détection du mouvement, le temps de collision, la structure, le centre de l'expansion et aussi la segmentation de l'objet. Le flot optique est généré par recouvrement du champ de 2-dimensions du mouvement. Par exemple la projection de la vélocité de 3-dimensions sur un plan de 2-dimensions. Le fait que le flot optique calcule la détection du mouvement est intéressant pour notre sujet de recherche. Nous allons nous appuyer sur le calcul du flot optique pour détecter le mouvement de la caméra.

Nous allons envisager dans un premier temps le champ du mouvement, puis nous parlerons du flot optique, et enfin des techniques pour calculer le flot optique.

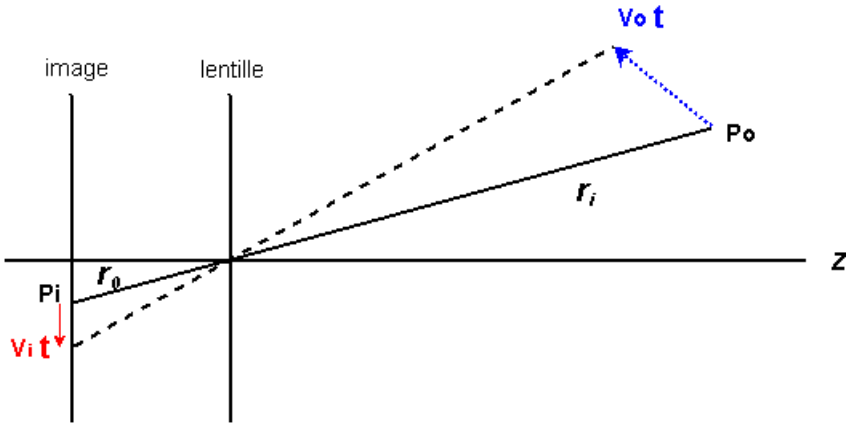

Figure 4 – 1 *Le mouvement 3-D de l'objet se projette sur l'image 2-D, pour créer le champ du mouvement*

4.2 Champ du mouvement et Flot optique

Quand un objet se déplace devant une caméra, il y a un changement dans l'image correspondante, comme le montre dans la figure 4-1. Donc si un point P_o de cet objet se déplace avec une vélocité $\mathbf{v}_o$, alors on peut affecter au point correspondant sur l'image P_i d'un vecteur de vélocité $\mathbf{v}_i$ en tant que sa vélocité du mouvement sur le plan de l'image. L'ensemble de toutes ces vélocités forme le champ du mouvement.

Si on ne s'occupe que des translations et rotations d'objet, le champ du mouvement peut être continu excepté les frontières des silhouettes des objets.

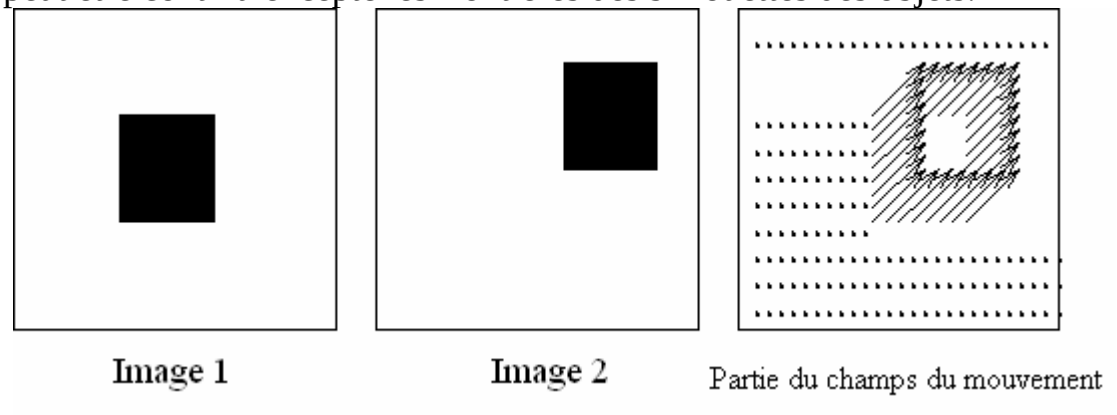

Figure 4 – 2 *Le champ du mouvement*

Pour un moment donné, il y a une surjection entre chaque point sur l'image et son point sur l'objet 3-D correspondant. Cette relation peut être obtenue par la formule de la projection. Avec la projection perspective, la ligne qui relie les deux points en passant par le point focal est appelée le rayon du point de l'image. Voir la figure 4-1.

Supposons qu'un point P_o sur l'objet a une vélocité relative par rapport à la caméra $\mathbf{v}_o$, donc le point projeté P_i sur l'image a donc une vélocité $\mathbf{v}_i$. Pendant le temps $\boldsymbol{\delta t}$ (t dans la Figure), le P_o se déplace $\mathbf{v}_o\boldsymbol{\delta t}$, et le point P_i se déplace $\mathbf{v}_i\boldsymbol{\delta t}$. La vélocité peut être présentée comme : (4 – 1)

$$\mathbf{v}_0 = \frac{d\mathbf{r}_0}{dt} \qquad \mathbf{v}_i = \frac{d\mathbf{r}_i}{dt}$$

où la relation entre $\mathbf{r}_0$ et $\mathbf{r}_i$ est : (4 – 2)

$$\frac{1}{f'}\mathbf{r}_i = \frac{1}{\mathbf{r}_0 \cdot \hat{\mathbf{z}}}\mathbf{r}_0$$

où f' représente la distance entre le plan d'image et le centre focal, $\hat{z}$ est l'unité scalaire de l'axe Z.

La formule (4-2) est utilisée simplement pour décrire la relation entre l'objet 3-D et le plan d'image, on s'intéresse plutôt au changement de luminosité afin d'avoir les informations sur la scène. Quand l'objet bouge, le pattern de luminosité de la partie correspondante de l'image change aussi. Donc le flot optique est le mouvement apparent des patterns de la luminosité dans l'image. Ici on utilise la notion 'mouvement apparent', c'est parce que le flot optique ne peut pas être décidé simple-

ment par le mouvement partiel de l'image. Par exemple les régions ayant la luminosité homogène ou ayant la luminosité sur une iso-ligne ne permettent pas de déterminer les mouvements des points, mais on peut quand même les observer, d'où un autre terme courant synonyme du flot optique est flot d'image.

Dans le cas idéal, le flot optique correspond au champ du mouvement, mais cela n'est pas toujours vrai. La figure 4-3 montre une sphère rotative sous le luminaire fixé. Puisque la surface de la sphère est une courbure, la luminosité apparaît un pattern de la distribution lumineuse et sombre (PDLS). Quand la sphère tourne de son axe central devant la caméra (à gauche de la figure ci-dessus), le PDLS ne change pas en fonction de la rotation de la sphère, donc l'image ne change pas non plus, alors le flot optique est égal à zéro partout. Mais le champ du mouvement n'est pas du tout égal à zéro. En revanche si la sphère reste immobile, et le luminaire se déplace autour de la sphère (à droite de la figure), le PDLS change en fonction du mouvement du luminaire. Dans ce cas-là pour la sphère le flot optique n'égale pas à zéro mais le champ du mouvement égale à zéro. Mais en général on peut supposer qu'il n'y a pas de grande différence entre le flot optique et le champ du mouvement, on peut donc estimer le mouvement en fonction du mouvement apparent dans l'image. C'est pourquoi on en parle dans ce mémoire, car avec les images capturées par la caméra, on peut directement calculer le mouvement de l'image afin de détecter le mouvement de la caméra en temps réel.

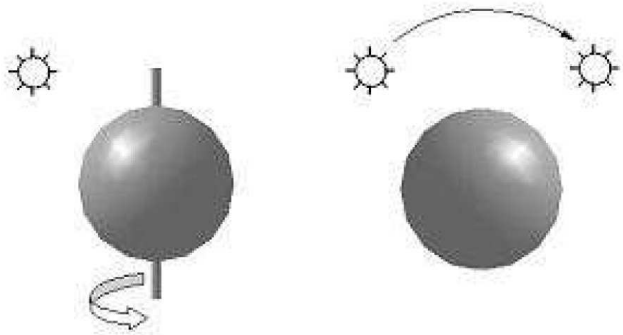

Figure 4 – 3 *Le champ du mouvement est différent que le champ du flot optique*

4.3 L'équation de contrainte de flot optique

Soit $I(x, y, t)$ est l'illumination du point (x, y) sur l'image au moment t .Une séquence d'images est une fonction réelle I(x, y, t) de trois variables t, x et y que nous supposons pour l'instant continues. Le modèle mathématique standard utilisé pour trouver des équations qui définissent le flot optique est basé sur une hypothèse d'illumination constante.

Si $u(x, y)$ et $v(x, y)$ sont les composants orientés sur l'axis x et l'axis y du vecteur du flot optique du point (x, y), le flot optique au temps t et au point (x, y) est alors défini comme la vitesse du point d'image

$$\upsilon = (u, v) = (\frac{d\delta x}{dt}, \frac{d\delta y}{dt}) \quad ,$$

où $\delta x = u\,\delta t$, $\delta y = v\,\delta t$.

Supposons que l'illumination soit constante quand ce point se déplace à la

coordonnée ($x + \delta x$, $y + \delta y$) au moment $t + \delta t$, Soit : (4 – 3)

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t)$$

Cette contrainte ne permet pas encore de résoudre u et v, donc on a besoin encore d'autre contrainte. Si la luminosité change continuellement selon x、y、t, on peut factoriser la partie gauche de l'équation 4-3 en degré de Taylor : (4 – 4)

$$I(x, y, t) + \delta x\frac{\partial I}{\partial x} + \delta y\frac{\partial I}{\partial y} + \delta t\frac{\partial I}{\partial t} + e = I(x, y, t)$$

où e est le terme de l'ordre de deux et supérieur de deux pour δx, δy, δt. Dans la formule les $I(x, y, t)$ de deux cotés s'annulent, et on divise les deux cotés par δt, puis on calcule la limitation de $\delta t \to 0$, c'est (4 – 5)

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

Et c'est en fait la factorisation de la formule suivante : (4 – 6)

$$\frac{dI}{dt} = 0$$

Soit :

$$I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y} \quad I_t = \frac{\partial I}{\partial t}$$

$$u = \frac{dx}{dt} \qquad v = \frac{dy}{dt}$$

Donc à partir de la formule (4 – 5) on obtient la relation entre le gradient spatio-temporal et le composant de la vélocité : (4 – 7)

$$I_x u + I_y v + I_t = 0$$

Ou bien : (4 – 8)

$$\nabla \mathbf{I} \cdot \mathbf{v} + I_t = 0$$

La formule ci-dessus est appelée l'équation de contrainte de flot optique. Les I_x, I_y et I_y sont calculables par l'image.

En fait cette équation de contrainte de flot optique génère le composant de la direction du gradient de l'intensité du mouvement de l'image ayant la luminosité constante, v_n. L'équation peut être écrite comme : $(\boldsymbol{I}_x, \boldsymbol{I}_y) \bullet (\boldsymbol{u}, \boldsymbol{v}) = -\boldsymbol{I}_t$,

Donc:

$$(u, v) = \frac{-I_t}{\sqrt{I_x^2 + I_y^2}}$$

$v_n = s\boldsymbol{n}$, où $\boldsymbol{n}$ et s sont respectivement la direction et la quantité de mouvement du gradient de l'intensité de l'image :

$$\mathbf{n} = \frac{\nabla \mathbf{I}}{\|\nabla \mathbf{I}\|} \qquad s = \frac{-I_t}{\|\nabla \mathbf{I}\|}$$

Sur l'image, il y a deux variables inconnues u et v sur chaque point, mais il n'y a qu'une équation, donc en utilisant simplement un point on ne peut pas calculer

le flot optique. Cela est appelé le problème d'ouverture. Voir la figure 4-4. Si on observe par une fenêtre (ouverture) le flot optique d'un cylindre publicitaire rotatif chez le coiffeur, on peut trouver qu'il n'est pas dans la même direction que le champ du mouvement.

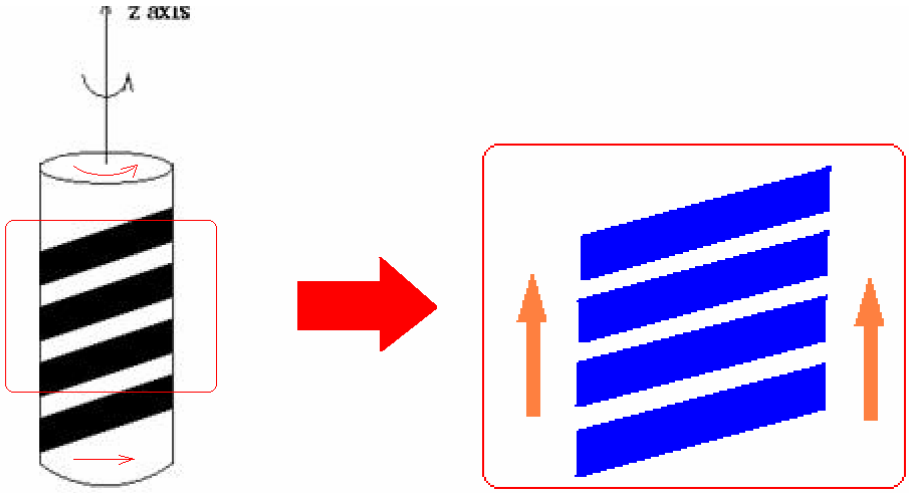

Figure 4 – 4 *Le flot optique observé par une ouverture n'est pas dans la même direction que le champ du mouvement*

En théorie on ne peut que détecter le mouvement du point de l'image selon l'orientation de gradient de l'intensité, ce qui correspond à la notion de flot normal. Supposer que la direction du mouvement de l'objet est **r**, voir la figure 4-5. Si on utilise une ouverture locale (O1) pour observer le mouvement, on ne peut pas déterminer si le mouvement de l'image longe le bord ou la perpendiculaire du bord. L'orientation de la perpendiculaire du bord est le flot normal. Mais si on observe par l'ouverture 2 (O2), on peut détecter le bon mouvement. C'est parce que par l'ouverture 2, l'image a le changement du gradient de l'intensité sur les deux perpendiculaires des deux bords. De cette manière on pourra peut-être estimer le mouvement de l'image sur un ensemble des pixels ayant assez de changement du niveau de gris. Ici on suppose que tous ces pixels ont le même vecteur de mouvement.

4.4 Calculs du flot optique

D'après ce qui précède, à cause du problème d'ouverture, on ne peut pas calculer le flot optique sur un certain point de l'image en utilisant l'équation de contrainte de flot optique toute seul. Donc on va voir dans cette partie plusieurs méthodes de calcule de flot optique permettant de résoudre le problème d'ouverture.

4.4.1 Méthode de Horn - Schunck

Cette méthode utilise l'hypothèse du lissage global de l'image. Soit le champ du mouvement satisfait à l'équation de contrainte de flot optique et aussi à la condition de la lisse globale. Le choix du terme de lisse minimise le gradient absolu de la vélocité :

$$\int [(\nabla I \cdot v + I_t)^2 + \lambda^2 (\left\|\nabla_2^2 u\right\| + \left\|\nabla_2^2 v\right\|)] d\vec{x}$$

Cette formule peut être réduite à une paire d'équations récursives, qui peuvent être résolues itérativement. Quand la différence entre les résultats de deux itérations adjacente est inférieure à un seuil, on arrête l'itération.

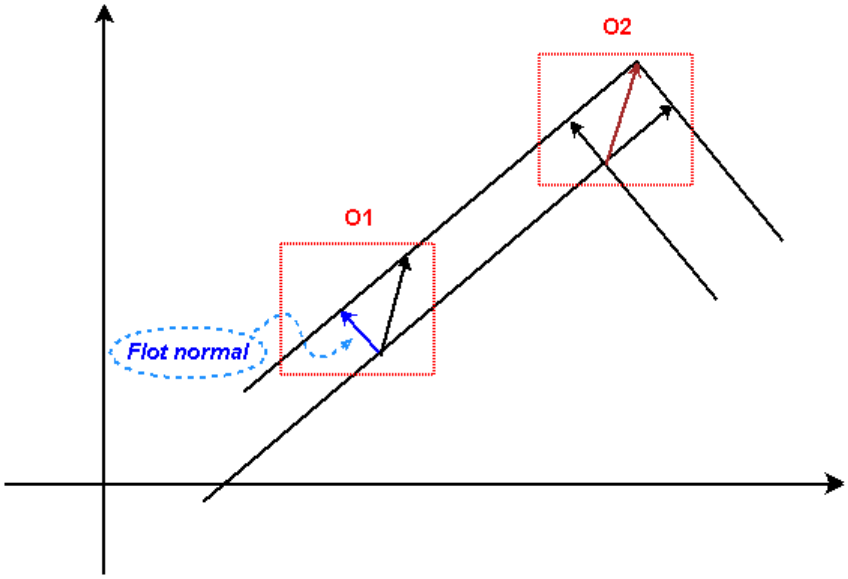

Figure 4 – 5 *Illustration du problème d'ouverture, avec deux fenêtres*

4.4.2 Méthode de Lucas - Kanade

Cette méthode suppose que le vecteur de mouvement reste constant sur un petit domaine spatial Ω, puis elle utilise des moindres carrés pondérés (weighted least-squares) pour estimer le flot optique. Contrairement à la régularisation du post-lissage de Horn-Schunk, elle choisit de post-lisser les données avant d'utiliser l'équation de contrainte de gradient (de flot optique). La minimisation des erreurs est :

$$\min \sum_{\vec{x}\in\Omega} W^2(\vec{x})\left[\nabla I(\vec{x},t)\cdot\vec{v} + I_t(\vec{x},t)\right]^2$$

où W($\vec{x}$) est la fenêtre qui a plus d'influence sur la contrainte autour du centre de Ω . Cela peut être réduit à une solution de format fermé pour l'estimation du flot, où :

$$\vec{v} = \left[A^T W^2 A\right]^{-1} A^T W\vec{b}$$

et

$$A = [\nabla I(\vec{x}_1),\ldots,\nabla I(\vec{x}_n)]^T$$
$$W = diag[W(\vec{x}_1),\ldots,W(\vec{x}_n)]$$
$$\vec{b} = -[I_t(\vec{x}_1),\ldots,I_t(\vec{x}_n)]^T$$

Un avantage important de cette méthode par rapport à celle de Horn-Schunk est l'existence de la mesure fiable. La valeur propre la plus petite, λ_1, de

$$A^T W^2 A = \begin{bmatrix} \sum W^2 I_x^2 & \sum W^2 I_x I_y \\ \sum W^2 I_x I_y & \sum W^2 I_y^2 \end{bmatrix}$$

fournit une mesure pour distinguer la vélocité normale à partir de la vélocité 2-D.

4.4.3 Méthode de Nagel

Nagel a utilisé les dérivés de second ordre pour estimer le flot optique, cela est une variante de la méthode Horn-Schunck. En plus, Nagel a utilisé également la contrainte de lissage global pour construire une fonction de mesure des erreurs de flot optique, qui est différente de celle de Horn-Schunck. Nagel a formulé une contrainte de lissage orientée, dans le but de traiter le problème de l'occlusion. Cette méthode a une fonction de mesure des erreurs qui est :

$$\iint \left(\nabla I^T \vec{v} + I_t\right)^2 + \frac{\alpha^2}{\|\nabla I\|_2^2 + 2\delta} \times \left[\left(u_x I_y - u_y I_x\right)^2 + \left(v_x I_y - v_y I_x\right)^2 + \delta\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right)\right] dxdy$$

où δ est une constante fixée.

Chapitre 5

L'Analyse de Bloc

5.1 Introduction

L'analyse du mouvement basée sur la notion de bloc est largement utilisée pour le traitement d'image et la détection du mouvement dans l'image. Par exemple dans le domaine de la technique de compression de vidéo numérique, la norme internationale MPEG1-2 adopte les algorithmes d'analyse et de compensation d'image basés sur cette notion. La détection du mouvement basée sur le bloc est différente que celle utilisant le flot optique, elle n'a pas besoin de traiter respectivement le mouvement de chaque pixel, mais elle calcule le mouvement de bloc qui est composé par plusieurs pixels. Pour des analyses et des détections d'image, l'analyse du mouvement basé sur la notion de bloc est une très bonne approximation.

5.2 Modèle du mouvement de bloc

Le modèle du mouvement basé sur un bloc suppose que le mouvement d'image peut être représenté par le mouvement de ce bloc. Le mouvement du bloc est en général classé en forme comme la translation, la rotation, le mouvement de transformation affine, le mouvement de la perspective, etc. En général le mouvement de bloc est une combinaison des formes, on l'appelle *mouvement déformé*.

5.2.1 La translation

Supposons que tous les blocs font la translation. On choisi un bloc **B** ayant le centre de la position (***x***, ***y***) dans l'image ***k***. Dans l'image ***k*** + 1, les relations entre tous les pixels et les niveaux de gris de tous les pixels restent invariants, et le centre du bloc se déplace à (***x′***, ***y′***). Donc la formule de la translation entre ***k*** et ***k*** + 1 est :

$$\begin{aligned} x' &= x + \Delta x \\ y' &= y + \Delta y \end{aligned} \tag{5.1}$$

Donc pour tous les pixels, on a donc :

$$f(x, y, t_k) = f(x + \Delta x, y + \Delta y, t_{k+1}) \quad (x, y) \in B \tag{5.2}$$

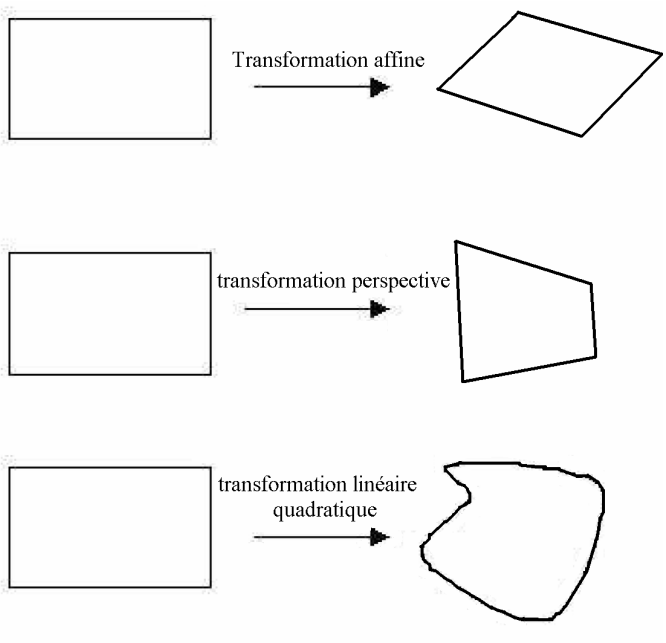

Figure 5-1 *Trois transformations spatiales*

5.2.2 Le mouvement de transformation affine

Nous allons faire une extension de la translation ci-dessus au mouvement de transformation affine :

$$\begin{aligned} x' &= a_{00}x + a_{01}y + a_{02} \\ y' &= a_{10}x + a_{11}y + a_{12} \end{aligned} \tag{5.3}$$

Cette formule peut décrire la translation, la rotation, et aussi des mouvements déformés de bloc illustrés à la figure 5-1. Ici la transformation affine désigne le mouvement 3-D d'un plan projeté parallèlement sur un plan d'image, cela a une propriété : deux lignes droites parallèles sont toujours parallèles après la transformation affine.

5.2.3 La transformation de projection perspective

$$\begin{aligned} x' &= \frac{a_{00}x + a_{01}y + a_{02}}{a_{20}x + a_{21}y + 1} \\ y' &= \frac{a_{10}x + a_{11}y + a_{12}}{a_{20}x + a_{21}y + 1} \end{aligned} \tag{5.4}$$

5.2.4 La transformation linéaire quadratique

$$\begin{aligned} x' &= a_{00}x + a_{01}y + a_{02}xy + a_{03} \\ y' &= a_{10}x + a_{11}y + a_{12}xy + a_{13} \end{aligned} \tag{5.5}$$

5.2.5 Bilan

Ces méthodes peuvent surmonter le problème d'ouverture. Elles n'ont pas besoin de calculer le mouvement de tous les pixels, mais le mouvement d'un bloc.

Elles sont plus simples et plus efficaces au niveau de calcul. Mais elles ne conviennent pas aux mouvements de rotations et de l'échelle d'image et non plus à la déformation partielle d'image.

5.3 Méthode de Fourrier

Avec la méthode de Fourrier on peut détecter et estimer la translation 2-D, la rotation et l'échelle de bloc.

5.3.1 Détection de translation

D'abord on applique la transformation de Fourrier sur la formule 5.2 :

$$F_k(\xi,\eta) = \Phi_k\{f(x,y,t_k)\} \tag{5.6}$$

$$\begin{aligned} F_{k+1}(\xi,\eta) &= \Phi_{k+1}\{f(x+\Delta x, y+\Delta y, t_{k+1})\} \\ &= \Phi_k\{f(x,y,t_k)\}\exp[-j2\pi(\xi\Delta x+\eta\Delta y)] \\ &= F_k(\xi,\eta)\exp[-2j\pi(\xi\Delta x+\eta\Delta y)] \end{aligned} \tag{5.7}$$

Aux moments t_k et t_{k+1}, la différence entre des phases 2-D de Fourrier des deux images est :

$$\begin{aligned} \Delta\varphi(\xi,\eta) &= \angle F_{k+1}(\xi,\eta) - \angle F_k(\xi,\eta) \\ &= -2\pi(\xi\Delta x+\eta\Delta y) \end{aligned} \tag{5.8}$$

La formule ci-dessus est en fait un plan défini par deux variable (ξ, η). L'estimation du vecteur du mouvement de bloc revient donc à calculer le (Δx, Δy) du sens de flot normal (notion de chapitre 4) de ce plan. Après la résolution d'équation on a Δx :

$$\Delta x = -\frac{\Delta\varphi_x(\xi)}{2\pi\xi}$$

Ainsi que Δy :

$$\Delta y = -\frac{\Delta\varphi_y(\eta)}{2\pi\eta}$$

Le (Δx, Δy) est le vecteur du mouvement du bloc d'image de moment t_k à t_{k+1} .

5.4 Méthode de l'appariement de bloc

Le concept de l'appariement de bloc est illustré par la figure 5-2. On choisit un bloc ***W*** de taille m **x** n avec un centre (*x*, *y*) dans l'image ***k***, et puis dans l'image ***k***+1 on définit une fenêtre plus grande du centre (*x*, *y*) pour chercher un bloc le mieux apparié du bloc ***W*** dans la fenêtre. Notre objectif est de trouver le vecteur de mouvement du centre du ***W*** entre deux image, donc **r** = (Δ*x*, Δ*y*). Il y a plusieurs algorithmes d'appariement, les différences entre eux sont représentées par les aspects suivants: règle de appariement, stratégie de recherche, choix de taille de la fenêtre.

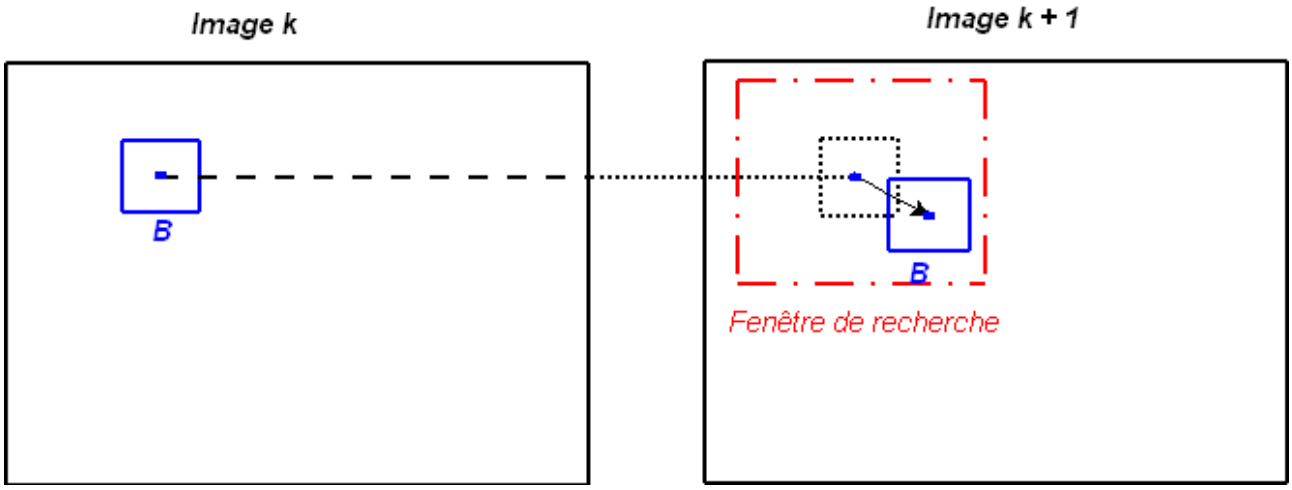

Figure 5-2 *Illustration de l'appariement de bloc*

5.4.1 Les règles d''appariement

Les règles existantes sont : relation maximum, erreur minimum de moyenne des carrés (mean square error) (MSE), différence absolue minimum de moyennes (mean absolute difference), comptage maximum de pixels d'appariement (matching pixel count) (MPC).

1) L'erreur minimum de moyenne des carrés est définie comme :

$$MSE(\Delta x, \Delta y) = \frac{1}{mn} \sum_{(x,y)\in W} [I(x, y, k) - I(x + \Delta x, y + \Delta y, k + 1)]^2$$

On calcule le minimum pour avoir le vecteur de mouvement $\mathbf{r} = (\Delta x, \Delta y)$, soit :

$$[\Delta x, \Delta y]^T = \arg \min_{(\Delta x, Xy)} MSE(\Delta x, \Delta y)$$

2) La différence absolue minimum de moyennes est définie comme :

$$MAD(\Delta x, \Delta y) = \frac{1}{mn} \sum_{(x,y)\in W} | I(x, y, k) - I(x + \Delta x, y + \Delta y, k + 1) |$$

Le vecteur du mouvement $\mathbf{r} = (\Delta x, \Delta y)$ est estimé :

$$[\Delta x, \Delta y]^T = \arg \min_{(\Delta x, Xy)} MAD(\Delta x, \Delta y)$$

3) La méthode du comptage maximum de pixels d'appariement classe les pixels dans la fenêtre les pixels selon leur appariement réussi ou échoué :

$$T(x, y, \Delta x, \Delta y) = \begin{cases} 1 & \text{Si} \quad | I(x, y, k) - I(x + \Delta x, y + \Delta y, k + 1) | \leq T \\ 0 & \text{Sinon} \end{cases}$$

T est un seuil prédéfini, donc le nombre maximum de pixels en appariement est :

$$MPC(\Delta x, \Delta y) = \sum_{(x,y)\in W} T(x + \Delta x, y + \Delta y)$$

$$[\Delta x, \Delta y]^T = \arg \min_{(\Delta x, Xy)} MPC(\Delta x, \Delta y)$$

5.4.2 Stratégie de recherche

Il existe une méthode de recherche complète, qui peut avoir la meilleur estimation, mais elle a besoin beaucoup de temps. D'autres méthodes plus rapides

sont donc proposées. On va présenter une recherche en n étapes.

Soit une fenêtre de 15 x 15 au centre on marque ‘0’, voir la Figure 5-3 (a).

Etape 1 : On prend les neuf pixels marqués ‘0’ et ‘1’ pour calculer la valeur de la fonction de la règle d’appariement. Si le meilleur appariement est au pixel ‘0’, cela veut dire qu’il n’y a pas de mouvement.

Etape 2 : On choisit 8 pixels (marqués ‘2’ dans la Figure) autour du pixel choisi à l’étape 1, on calcule la valeur de fonction de la règle d’appariement.

Etape 3 : On choisit 8 pixels (marqués ‘3’ dans la Figure) autour du pixel ayant la meilleure valeur dans l’étape 2. On calcule la valeur de fonction de la règle d’appariement. La meilleure valeur désigne la meilleure estimation.

En conséquence on voit bien que la surface de la fenêtre de recherche se réduit à une moitié, et que l’estimation est de plus en plus proche vers l’estimation exacte. Normalement on fait au moins 3 étapes, et selon le besoin on peut faire plus de 3 étapes pour avoir plus de précision, c’est la raison pour laquelle cette méthode est dite *recherche à n étapes*.

Il existe d’autres voies de recherche, voir la Figure 5-3 (b). A chaque étape on recherche sur 4 pixels répartis dans une position de forme de croix, on effectue la recherche itérativement comme la recherche à n étapes.

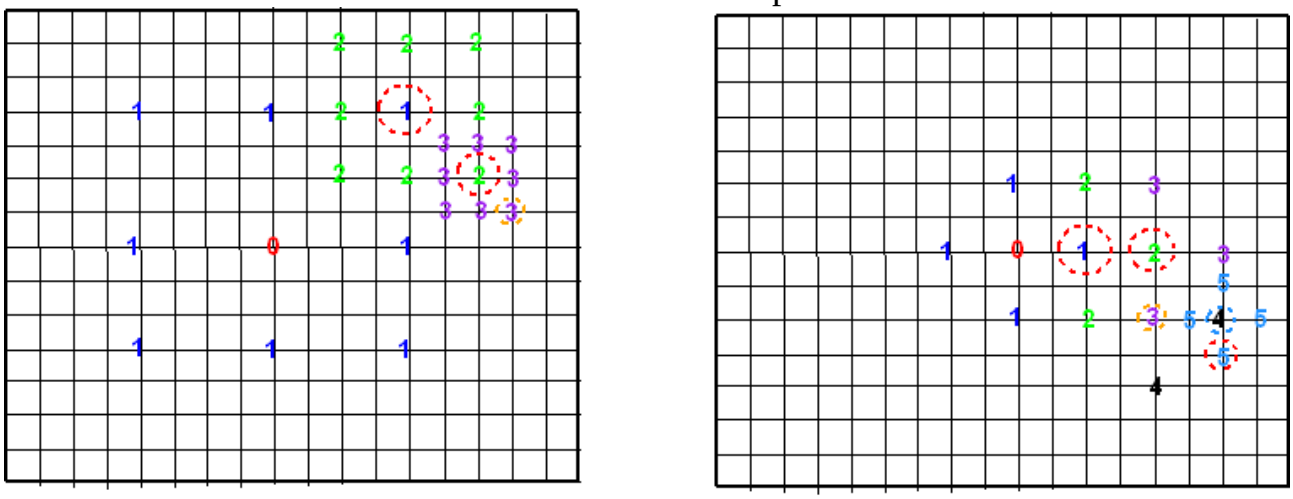

Figure 5-3 *Méthode de la recherche du bloc correspondant*

En pratique, le [21] utilise l’algorithme de l’appariement de bloc hiérarchique pour estimer le mouvement propre d’une caméra. Voir la figure ci-dessous.

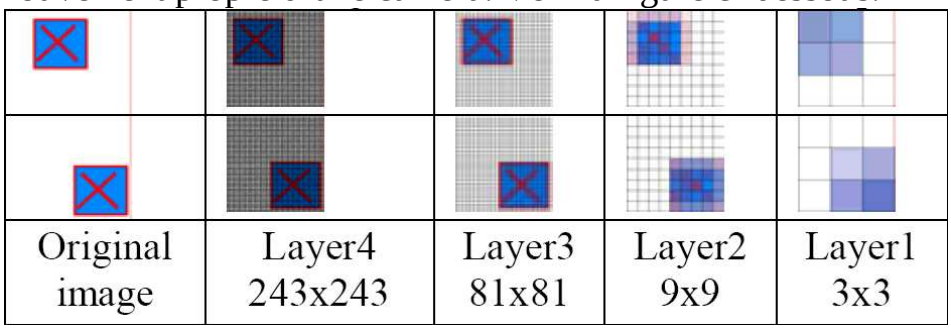

Figure 5-4 *L’appariement hiérarchique*

Elle représente un appariement de niveau grossier au niveau fin.

Chapitre 6

L'Implémentation

Dans ce chapitre nous allons présenter la partie des développements de cette recherche.

Nous allons d'abord présenter une expérimentation de détecter le mouvement en nous appuyant sur l'appariement de bloc.

Ensuite une expérimentation de la détection du mouvement de la caméra qui filme une feuille à carreaux en se déplaçant.

Enfin nous allons décrire une expérimentation primaire de détection du mouvement d'une caméra de téléphone portable.

6.1 Appariement de bloc

Dans cette partie nous présentons un développement réalisé sur la base de la conception de l'appariement de bloc. C'est un développement heuristique, car il est très encadré et idéalisé. Ce qui veut dire que la suite d'images capturées par la caméra est créée artificiellement. Donc le mouvement entre les images adjacentes ne concerne que la translation. Les images ne doivent pas avoir d'effet de rotation ni de changement de luminosité. La distance entre les objets et la caméra est maintenue invariable, l'apparence des objets dans les différentes images reste la même. En résumé, tous les objets (vues partielles) sont représentés par les mêmes combinaisons de couleurs dans les différentes zones des images. Par exemple une bibliothèque chargée des livres, quand on bouge la caméra elle apparaît de différentes positions dans les images, mais on rassure que l'ensemble de pixels représentant la bibliothèque soit identiques. En fait dans le réel cela n'est pas possible, on dit donc que ce développement ici est très normatif et idéal.

L'objectif de ce développement est de détecter le mouvement de caméra entre chaque deux moments de capturer deux images adjacentes. En s'inspirant de la méthode de l'appariement de bloc au Chapitre « analyse de bloc », on pourra s'entraîner à détecter le mouvement de translation. Pour faire cela il faut d'abord choisir un bloc de l'image précédente, et puis chercher l'emplacement de ce bloc à

l'image suivante en calculant le vecteur de mouvement apparent du bloc dans les deux images, on pourra ainsi obtenir le mouvement de la caméra.

Avant de commencer à présenter notre algorithme d'appariement, la construction de la suite d'images idéale est présentée. En fait pour répondre à toutes ces contraintes, le seul moyen est de faire en sorte que les mêmes objets dans différents plans proviennent d'une même image initiale afin de rassurer que les mêmes objets apparaissent avec les pixels correspondants de même couleur dans les différentes images. En conséquence on choisit une image assez grande pour la découper, puis on extrait la suite d'image. Voir la Figure 6-1.

Figure 6-1 *Extraction d'une suite d'images idéales*

Nous allons ensuite essayer de détecter le mouvement de la caméra. Pour faire cela il faut calculer le mouvement apparent représenté par les objets dans les images. D'après la méthode de l'appariement, il faut d'abord définit un bloc dans l'image précédente, puis le rechercher dans l'image suivante. Ici on fait à l'inverse. On utilise toujours les blocs au bord de l'image actuelle, et puis on en retrouve la plus grande partie dans l'image précédente. Voir la Figure 6-2. Au premier rang de la Figure, c'est l'image précédente et l'image actuelle. Nous constatons qu'une partie des pixels (rouge) de l'image précédente occupe une partie de l'image actuelle. On définit dans l'image actuelle un bloc B dont la taille est de largeur de l'image et de la hauteur, d'un quart de la hauteur de l'image, il est encadré par un rectangle vert à droit au premier rang à la Figure. Dans le bloc B on voit qu'il y a une partie rouge B_{com} et une partie bleue, le travail consiste à trouver le B_{com} dans l'image précédente. La Figure montre également comment on le recherche. Donc le B_{com} va s'apparier avec le premier B_{prcd} qui est à la première ligne de l'image précédente, et il va trouver une la

plus grande partie commune avec le B_{prcd} de l'image précédente. Cela peut se faire en décalage par colonne. Et puis en effectuant cela itérativement pour toutes les lignes on peut aussi déterminer une valeur de la note (nombre minimum de décalage par colonne) d'appariement pour chaque ligne. On prend la ligne qui ont la meilleure note (minimum), cela désigne la plus grande partie commune. On peut calculer la coordonnée du B_{com} dans l'image précédent. Cela est illustré à droite en bas de la Figure 6-2, le B est placé au rectangle jaune dans l'image précédent.

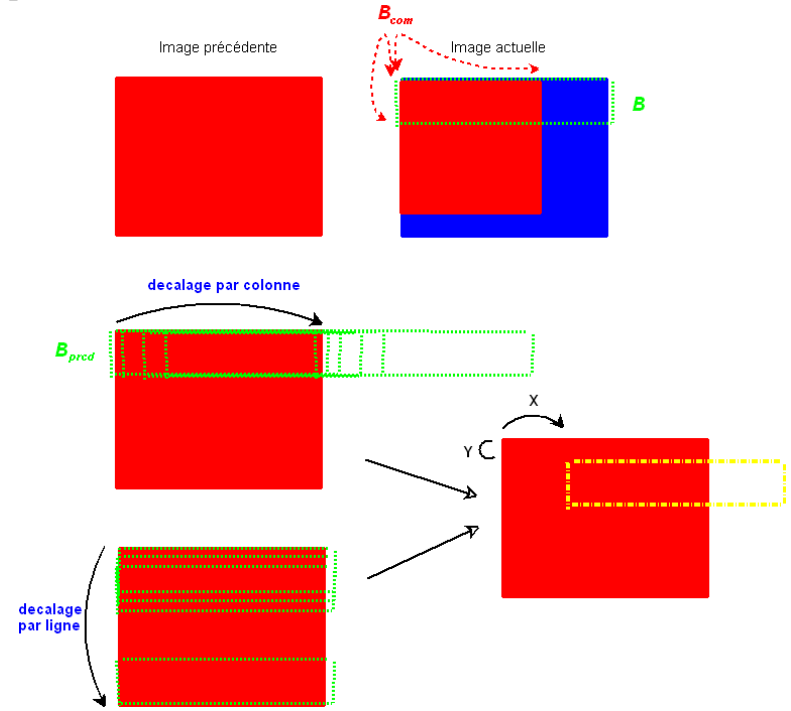

Figure 6-2 *Définition du bloc et de la façon de recherche*

En fait, le vecteur de mouvement peut avoir 4 possibilités au niveau de l'orientation. Les 4 cas peuvent se déduire grâce aux quatre zones de la coordonnée cartésienne. Soit : (x>0, y>0), (x<0, y>0), (x<0, y<0) et (x>0, y<0). La Figure 6-3 illustre comment la partie commune se place dans l'image précédente pour les 4 cas.

Donc à chaque fois on doit prendre deux blocs B (en haut et en bas) dans l'image actuelle, car on ne sait pas quel bloc est le bloc inutile. Et pour chaque bloc on doit l'apparier dans l'image précédente en commençant par gauche et aussi par la droite, cela est parce que la partie commune B_{com} peut être un sous-bloc de droite ou de gauche du bloc B.

L'algorithme 6-1 ci-après est pour trouver le mouvement apparent du bloc commun dans deux images.

La figure 6-4 illustre un exemple sur deux images adjacentes sélectionnées de la suite d'images de la Figure 6-1. L'image à gauche est la précédente, l'actuelle est à droite. On a choisi un bloc en haut ***Bh*** et un autre en bas ***Bb*** dans l'image actuelle. Le ***Bh*** a trouvé dans l'image précédente, un bloc correspondant plus petit que le bloc correspondant trouvé par le ***Bb***. D'après l'observation, on peut dire que dans l'image précédente, le bloc correspondant trouvé par le ***Bh*** presque n'existe pas, donc on l'appelle bloc inutile. Dans l'image précédente, le sous-bloc encadré en rouge est la partie commune pour le ***Bb*** dans l'image actuelle.

Algorithme 6-1 :

Créer une matrice (X * Y) pour image

Créer deux matrices ($\frac{X}{4}$ * Y) en haut et en bas de l'image courante pour les blocs ***B***.

Pour ligne = [0, $\frac{3}{4}X$) /* de 0 à $\frac{3}{4}X$, mais exclure $\frac{3}{4}X$ */

Pour colonne = [0, X) /* exclure X */

Comparer le bloc

Si c'est apparié, noter colonne, arrêter la boucle.

Fin pour

Fin pour

Noter quelle ligne a la plus grande partie commune, puis sur quelle colonne.

Choisir dans les deux blocs ***B*** un bloc qui a la plus grande partie commune.

Stocker la coordonné du bloc dans l'image précédente.(x, y)

Fin Créer

Calculer l'angle , a = ctan(x, y)

Fin Créer.

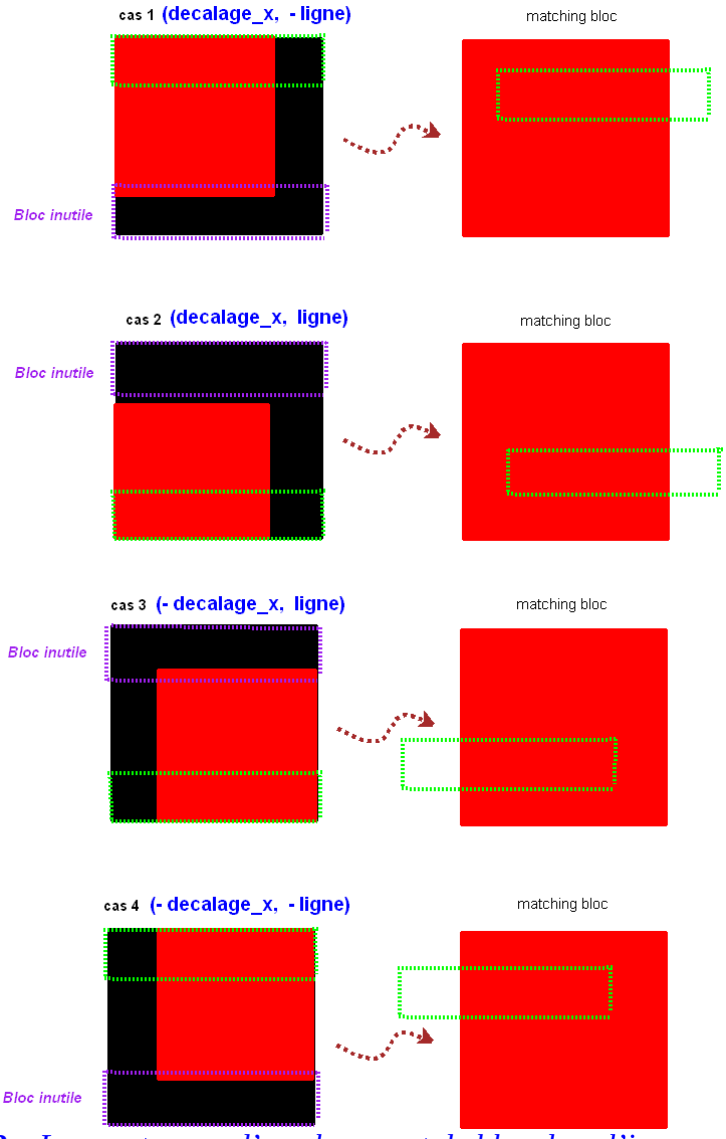

Figure 6-3 *Les quatre cas d'emplacement de bloc dans l'image précédente.*

Figure 6-4 *Exemple sur deux images adjacentes*

6.2 Simulateur de souris optique

Dans cette expérimentation, on réalise un simulateur de souris optique, cela veut dire qu'une caméra se déplace en filmant dans une zone définie, et d'après le film on reproduit la trajectoire de son mouvement. Pour faire cela, on prend une feuille à carreaux définie dans le Chapitre « suivi d'objet » comme une feuille blanche avec un petit carreau rouge au milieu. On peut considérer que la feuille blanche est l'arrière-plan, et le carreau rouge est l'objet que l'on veut suivre par la méthode de suivi d'objet. Quand la caméra se déplace, la vue de la caméra peut être considérée approximativement comme si le petit carreau rouge se déplace devant un arrière-plan constant soit la feuille blanche.

D'après la méthode dite soustraction de l'arrière-plan du suivi d'objet, on prend une photo de la feuille blanche comme l'arrière-plan, et puis fait la soustraction entre l'arrière-plan et les images capturées. Là où il y a des pixels qui ont leur valeur de la soustraction supérieure à un seuil prédéfini, nous avons les emplacements des pixels du carreau. De cette manière on peut suivre le carreau.

Une fois que l'on a le mouvement relatif du carreau, on peut à l'inverse détecter le mouvement de la caméra.

Le champ de la caméra ne doit pas déborder de la feuille, si non on perd l'arrière-plan.

On a fait une expérimentation d'abord sur une suite d'images capturées par la caméra, ensuite sur un morceau de film.

6.3 SourisPhone

La dernière expérimentation consiste à réaliser la deuxième expérimentation en partant d'un téléphone portable. Cela veut dire que l'on prend un téléphone portable avec une caméra embarquée. En activant la caméra on déplace le téléphone devant la feuille à carreaux, l'ordinateur affiche le mouvement de téléphone en temps réel. Dans ce cas-là, le téléphone marche comme une souris, donc on l'appelle donc un **SourisPhone**.

L'architecture de ce développement est illustrée dans la figure 6-5.

Le coté téléphone est codé en J2ME (Java 2 Micro Edition), On utilise le MMAPI (Mobile Media API) pour piloter la caméra.

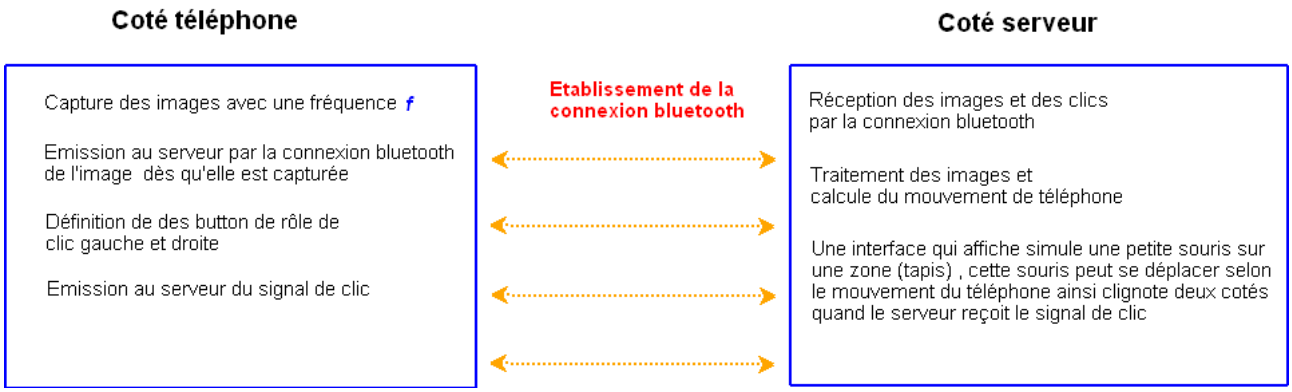

Figure 6-5 *L'architecture de développement de sourisphone*

Conclusion

L'étude décrite dans ce mémoire a exploré plusieurs aspects de la viabilité de la localisation et de la détection des mouvements sur un téléphone équipé d'une caméra. Les méthodes sont systématiquement comparées ci-dessous.

Le suivi d'objet est opérant pour détecter le mouvement produit par la caméra dont la vue contient toujours de mêmes objets en mouvement. Cela est pour but de limiter la zone du mouvement de caméra pour que le même objet soit toujours présent pendant toute la prise de vue de la caméra. Par exemple, pour détecter le mouvement de la caméra devant une feuille à carreaux, le mouvement est limité. Alors on fait la simulation du fonctionnement de la souris optique. Une solution pour agrandir la zone du mouvement est d'utiliser un plan contenant plusieurs particules définies dans le Chapitre 1.

La stabilisation d'image est une technique pour trouver un meilleur sens pour compenser l'image floue. Ceci est peut-être un long processus, parce que l'on essaie de différents sens pour trouver le sommet de la note de compensation. Donc il concerne des algorithmes pour rapprocher plus rapidement vers le sommet. Mais si on veut trouver le mouvement après avoir compensé l'image, il semble un tour inutile. Donc certains calculent d'abord le mouvement de l'image grâce au calcule de flot optique avant de compenser.

Le calcule de flot optique est une technique assez précise. Il calcule directement le mouvement apparent sur l'image pour détecter le mouvement, cela ne dépend ni de l'objet ni des images précédentes ou suivantes. On peut dire que cette technique a moins de contraintes. Mais le coût du calcule de flot optique est assez important, car il s'agit en effet du calcule de mouvement apparent de chaque pixel. De plus quelque fois on ne peut pas parfois tout à fait compter sur le flot optique, parce que le flot optique peut être différent que le champ de mouvement à cause du problème d'ouverture.

Enfin on arrive à l'analyse de bloc. Cette est basée sur le mouvement de bloc, ou l'ensemble de pixel, elle est donc moins coûteuse que le calcule du flot optique. Comme il s'agit de trouver dans l'image suivante un bloc qui est

correspondant à celui de l’image précédente, il peut donc avoir la confusion s’il y a plusieurs blocs approximatifs.

La conclusion de notre étude préconise est que la recherche concernant la localisation, gestuelles sur téléphone avec caméra est viable.

Références :

[1] **Makarov, A.** « *Comparison of Background Extraction Based Intrusion Detection Algorithms* » ***IEEE Int. Conf. Image Processing*** **(1996)**

[2] **Kehtarnavaz, N., Rajkotwala, F.** « *Real-Time Vision-Based Detection of Waiting Pedestrians* » **Real-Time Imaging, Vol. 3 (1997)**

[3] **J. Owens, A. Hunter and E. Fletcher.** « *A Fast Model-Free Morphology-Based Object Tracking Algorithm* » **University of Sunderland, BMVC (2002)**

[4] **Feng Tang and Hai Tao.** « *Object Tracking with Dynamic Feature Graph* » **Department of Computer Engineering, University of California, Santa Cruz**

[5] **R.T. Collins and Y. Liu.** « *On-Line Selection of Discriminative Tracking features* » ICCV '03, Nice, France. October, pp346-352 (2003).

[6] **Alberto Censi, Andrea Fusiello, Vito Roberto** « *Image Stabilization by Features Tracking* » **Machine Vision Laboratory, Dept. of Mathematics and Informatics, University of Udine, Italy**

[7] **S.-J. Choi, R. R. Schultz, R. L. Stevenson, Y.-F. Huang, and R.-W. Liu.** « *Contrast enhancement of missile video sequences via image stabilization and product correlation* » **University of Notre Dame, Department of Electrical Engineering, Laboratory for Image and Signal Analysis,1994.**

[8] **C. Morimoto and R. Chellappa. «** *Fast 3D stabilization and mosaic construction*» **In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 660–665, 1997.**

[9] **L. Wixson, J. Eledath, M. Hansen, R. Mandelbaum, and D. Mishra. «** *Image alignment for precise camera fixation and aim* **». In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 594–600, 1998.**

[10] **M. Irani, B. Rousso, and S. Peleg.** *« Recovery of ego-motion using image-stabilization »*. **Technical report, The Hebrew Universilty of Jerusalem, Institute of Computer Science, August 1993.**

[11] **L. S. Shapiro, H. Wang, and J. M. Brady. «** *A matching and tracking strategy for independently moving objects* **». In *Proceedings of the British Machine Vision Conference*, pages 306–315. BMVA Press, 1992.**

[12] **N. Gracias and J. Santos-Victor.** « *Automatic mosaics creation of the ocean floor* » **In *Proceedings of the OCEANS Conference*, 1998.**

[13] **Y. Seo, S. Choi, H. Kim, and K.-S. Hong.** «*Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick* ». **In *Proceedings of the International Conference on Image Analysis and Processing*, pages 196–203, 1997.**

[14] **H. Singh, J. Howland, and D. Yoerger.** « *Quantitative photomosaicking of underwater imagery* ». **In *Proceedings of the OCEANS Conference*, 1998.**

[15] **I. Zoghlami, O. Faugeras, and R. Deriche.** « *Using geometric corners to build a 2D mosaic from a set of images* ». **In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 420–425, 1997.**

[16] **A. Gelb, editor**. « *Applied Optimal Estimation* ». **The M.I.T. Press, 1974.**

[17] **Chen, Shaokang and Lovell, Brian C. (2001)** « *Real-Time MMX-Accelerated Image Stabilization System* » **In *IVCNZ2001*, 26-28 November, 2001, pages 163-168, Dunedin, New Zealand.**

[18] **Jesse S. Jin, Zhigang Zhu and Guangyou Xu** « *A Stable Vision System for Moving Vehicles* » ***IEEE Transactions on Intelligent Transportation Systems*, Vol.1, NO.1, pp. 32 – 38, March 2000 .**

[19] **Joon Ki Paik, Yong Chul Park and Sung Wook Park** «*An Edge Detection approach to Digital Image Stabilization based on Tri-state adaptive linear neurons* » ***IEEE Transactions on Consumer Electronics*, Vol.37, NO.3, pp.521 – 524, August 1999.**

[20] **Joon Ki Paik, Yong Chul Park and Wook Kim** « *An adaptive Motion Decision System for Digital Image Stabiliser Based on Edge Pattern Matching* » ***IEEE*** Transactions on Consumer Electronics, Vol. 38, NO.3, pp. 607 – 615, August 1992.

[21] **Xu Liu1, David Doermann, Huiping Li** *«FAST CAMERA MOTION ESTIMATION FOR HAND-HELD DEVICES AND APPLICATIONS »* **Applied Media Analysis, Inc.**

[22] **Nishkam Ravi, Pravin Shankar, Andrew Frankel, Ahmed Elgammal and Liviu Iftode,** « *Indoor Localization Using Camera Phones* » **Department of Computer Science, Rutgers University, Piscataway, NJ 08854**

[23] **Feng Tang and Hai Tao** *«Object Tracking with Dynamic Feature Graph»* **Department of Computer Engineering, University of California, Santa Cruz.**