Suivi d'objets en imagerie aérienne

Aude Jacquot

École doctorale INP Grenoble

Suivi d'objets en imagerie aérienne

THÈSE

présentée et soutenue publiquement le 2006

pour l'obtention du grade de

Docteur de l'INPG

Spécialité Imagerie, Vision et Robotique

par

Aude Jacquot

Composition du jury

Président :	Le président
Rapporteurs :	Le rapporteur 1 Le rapporteur 2 Le rapporteur 3
Examinateurs :	L'examinateur 1 L'examinateur 2

Ecole Doctorale Mathématiques, Informatique, Sciences et Technologie de l'Information

Mis en page avec la classe thloria.

Remerciements

Les remerciements.... à compléter

Peter Sturm et Olivier Ruch mes deux directeurs de thèse
Clément Ménier et Jean Sébastien Franco pour m'avoir supporté au jour le jour
Toute ma famille
Antoine bien évidemment

Mon jury...

La rigueur vient toujours à bout de l'obstacle.
Léonard de Vinci

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les progrès de l'informatique en terme de puissance de calcul permettent d'envisager aujourd'hui d'embarquer sur n'importe quel type de véhicule (terrestre, sous marin ou aérien) des fonctions sophistiquées de traitement d'images avec des performances en temps réel. Cette capacité de traitement ouvre de larges perspectives d'applications dans différents domaines tels l'analyse de scène, la modélisation d'objets, ou encore en robotique, pour des systèmes utilisant de l'information vidéo dans des boucles de commande. En termes d'applications de cette technologie, on trouve l'aide à la conduite ou la conduite automatique de véhicule, le guidage de robots, la vidéo surveillance, mais aussi les applications en réalité virtuelle et en interaction homme/machine pour les jeux vidéos.

La vision par ordinateur est un domaine vaste qui met en jeu des compétences en mathématiques aussi bien fondamentales qu'appliquées, intelligence artificielle, traitement du signal, automatique et informatique. Et même si aujourd'hui la théorie est parfois bien claire et formulée, il reste encore de nombreux défis pour faire fonctionner ces méthodes en pratique.

Le problème des performances des systèmes de vision aujourd'hui vient de la difficulté à extraire et à associer les informations pertinentes des images pour une application donnée. Il existe des informations qu'il est difficile de faire émerger parce qu'on ne les a pas identifiées ou parce que l'on n'arrive pas à les caractériser ou à les associer suffisamment bien ; ces informations sont comme dissimulées dans les images que l'on observe. Nous allons mettre en évidence ce problème d'émergence d'information en étudiant deux exemples : le premier lié à la reconnaissance de visages, et le second aux informations 3D.

La vision humaine est très performante pour la reconnaissance de visages. En apercevant une personne pour la deuxième fois, nous sommes en général capables d'affirmer que nous l'avons déjà vue. Cette reconnaissance implique l'existence d'*indices visuels* invariants. Cependant, les systèmes artificiels de reconnaissance de visages n'obtiennent pas encore des résultats du même ordre, ce qui montre que ces indices sont relativement complexes. Le problème est donc de déterminer quelles sont les informations qui correspondent à ces indices visuels invariants, et d'élaborer une stratégie pour les extraire des images. Sans rentrer dans les détails, essayons d'analyser un peu la nature de ces indices visuels. On sait que notre système de vision est capable de reconnaître des visages uniquement à partir d'une ébauche de traits caractéristiques ; les portraits robots par exemple, ou encore les caricatures de personnages célèbres ne sont qu'un ensemble de traits noirs sur du papier blanc mais suffisent pour identifier correctement un visage. Le problème ne consiste plus à caractériser les indices à extraire (on sait que les traits du visage sont les indices à extraire), mais plutôt à associer ces informations qui caractérisent les traits du visage. De nombreux travaux ont été tentés dans ce domaine, en décomposant les traits en courbes ou en segments, puis en les associant selon des symétries détéctées, mais les résultats ne sont pas

toujours convaincants. On peut, par exemple se référer aux travaux de Harmon [HKRR81] pour la reconnaissance de profils. Pourtant, on peut affirmer que certaines associations d'informations relatives à ces courbes devraient être suffisantes pour résoudre le problème. Ce sont ces informations associatives qui sont dissimulées derrière une description simple des contours qui n'émergent pas des algorithmes existants. En général, on étudie uniquement des relations binaires, alors que les neurones de notre cerveau sont souvent connectés à des centaines d'autres neurones. Cette remarque montre clairement l'ampleur du problème de l'*émergence de l'information*.

Un autre exemple concerne l'émergence des informations 3D : à partir d'un simple dessin sur une feuille de papier, c'est-à-dire sans aucune information sur la position en profondeur des objets de la scène, notre perception visuelle nous permet de nous représenter la scène en trois dimensions. Et même si cette représentation n'est pas exacte, elle nous permet d'avoir des indices visuels 3D de grande qualité. Si ensuite nous devions faire un travail de reconnaissance, la mise en correspondance sera grandement facilitée car les contraintes 3D sont telles que le résultat est quasi-immédiat et d'une fiabilité exceptionnelle malgré le degré d'incertitude des informations. Comme Biederman l'a montré avec sa théorie "Recognition by Components" [Bie85], la seule connaissance des contours détectés dans l'image, même entachés d'erreurs, permet à l'œil humain de reconnaître et de localiser correctement les objets de la scène dans la plupart des cas. De nombreuses heuristiques ont été formulées pour reconstruire la troisième dimension à partir d'une image. Parmi les heuristiques les plus intuitives, on trouve les trois suivantes :

- Si la caméra est posée sur un plan horizontal, les segments verticaux dans l'image sont généralement verticaux dans la scène.
- Un segment relié à deux segments verticaux est généralement horizontal. Son orientation 2D permet alors de remonter à son orientation 3D.
- Un segment à peu près parallèle à un autre possède en général la même orientation.

Ces heuristiques (non exhaustives ici) permettent de générer un grand nombre d'hypothèses sur la scène observée. L'ensemble de ces hypothèses génèrent un certain nombre de contraintes qui valident ou invalident les autres hypothèses, et en pratique il se dégage une unique représentation 3D satisfaisante. Cette solution ne peut pas être sure puisqu'aucune information 3D n'a été utilisée ; cependant les contraintes sur le monde réel sont tellement fortes qu'en pratique cette description 3D de la scène est tout à fait fiable. C'est d'ailleurs ce que nous confirme la vision humaine, puisque sans exploiter le mouvement, en fermant un œil, nous sommes toujours capables de décrire la scène en trois dimensions avec une qualité de reconnaissance exceptionnelle. Il existe donc là aussi, à travers cet exemple, des informations dissimulées derrière notre description trop simpliste de la scène, qui sont difficiles à faire émerger, et qui pourtant, si elles étaient prises en compte, permettraient d'obtenir une description fiable en trois dimensions de la scène.

Ce dernier exemple place le cadre de ces travaux de thèse : nous souhaitons utiliser la faisabilité d'une extraction d'information 3D à partir de séquences vidéo afin d'améliorer les performances en suivi de matériels aéroportés existants. Cette thèse résulte d'une collaboration CIFRE entre Thalès Optronique S.A et l'INRIA Rhône-Alpes.

La convention CIFRE, une collaboration entre un industriel et un laboratoire de recherche :

Spécialiste de l'optronique et de l'imagerie thermique, la société Thales Optronique SA (groupe Thales) conçoit et développe des systèmes de désignation aéroportés (appelés Pods) pour des "armements guidés laser" (AGL). Leader Européen pour ces équipements qui ont montré leur importance dans les récents conflits, TOSA mène une politique active de recherche et développement afin de maintenir son avance technologique. Notamment, TOSA cherche à doter ses futurs matériels de nouvelles fonctionnalités améliorant la précision et la portée de visée ainsi que la facilité d'emploi de ses équipements qui sont des avantages concurrentiels essentiels dans le domaine militaire. C'est dans ce cadre que s'inscrit ce travail de recherche.

L'INRIA Rhône-Alpes est une des six unités de recherche de l'INRIA, localisée à Montbonnot, près de Grenoble. Dans ce cadre, une grande partie des travaux de recherches ont été réalisés au sein du projet MOVI (MOdélisation, localisation, reconnaissance et interprétation en VIsion par ordinateur), projet de recherche internationalement reconnu par ses travaux scientifiques dans le domaine de la vision par ordinateur et ses applications. Les domaines de compétences des membres du projet sont ceux de la vision multi-caméras (théorie, analyse numérique, algorithmes et systèmes), de la reconstruction tridimensionnelle de points et de surfaces (par des méthodes de stéréoscopie, reconstruction voxellique ou enveloppes visuelles), de la calibration des systèmes de vision, analyse d'images et de vidéos par leur contenu (reconnaissance et détection d'objets, indexation), du suivi spatio-temporel d'objets et de personnes, du rendu de textures à partir de plusieurs vues, analyse du comprtement humain (gestes sportifs, mouvement des mains), et des méthodes statistiques pour l'apprentissage visuel.

La problématique des travaux :

Thales Optronique conçoit et développe des systèmes de désignation aéroportés pour des "armements guidés laser". Ces systèmes d'armes ont montré leur efficacité dans les récents conflits, en permettant de réaliser des interventions ciblées minimisant les dégâts collatéraux. Cette précision est atteinte grâce à l'emploi d'armements guidés par un faisceau laser éclairant la cible durant toute la phase d'approche du missile. Ce mode opératoire requiert de la part du pilote une charge de travail importante que ce soit avant ou pendant la mission.

Avant la mission, durant la phase de préparation de mission, la cible est sélectionnée et son contexte est caractérisé à partir de données issues de cartes géographiques ou de reconnaissance aérienne afin de fournir au pilote les éléments lui permettant de la reconnaître et de l'identifier précisément.

Durant la mission, on distingue deux phases :

1. La phase de désignation : avant l'éclairement de la cible, le pilote doit à partir de l'image fournie par sa caméra embarquée, reconnaître sa cible et déterminer sa position exacte dans l'image.
2. La phase de poursuite : la localisation de la cible étant effectuée, des "trackers" sont initialisés. Leur but est d'asservir le faisceau laser sur la cible durant le vol d'approche. Pour réaliser cet asservissement, le laser ainsi que la caméra sont montés sur une plate-forme gyrostabilisée possédant trois degrés de liberté en rotation. Pendant toute cette phase

d'éclairement de la cible, le pilote doit contrôler et éventuellement compenser la dérive des trackers afin d'assurer la précision de visée maximale.

L'objectif opérationnel des travaux est d'alléger la charge de travail demandée au pilote dans les phases de vol (désignation ou tracking) qui s'effectuent dans un contexte particulièrement stressant. Le problème opérationnel à résoudre est double :

1. D'une part, il s'agit d'automatiser la phase de désignation manuelle afin qu'elle se réduise à une phase de confirmation de l'identification de la cible par le pilote.
2. D'autre part, il consiste à rendre les trackers actuels plus robustes face à des effets de perspective, notamment en présence de reliefs.

Dans ces travaux, nous nous concentrons sur le deuxième objectif, celui d'améliorer le suivi, en prenant en compte de l'information 3D extraite des séquences vidéo. Le problème est d'être capable de suivre la cible dans la séquence d'images en compensant le mouvement de l'avion qui entraine des grands changements quant à la résolution de l'image et quant à l'apparence de la cible dans l'image.

En effet, le porteur se déplace à environ 300 mètres/seconde (1080 km/h), la cible apparait dans la séquence vidéo à des résolutions très différentes au fur et à mesure que l'avion s'en approche, d'un simple blob de quelques pixels à un objet 3D complexe. Le suivi doit tenir compte à la fois de la trajectoire du porteur et de ce changement de résolution. Le problème de suivi d'un objet dont l'apparence change d'une simple région à une structure 3D complète n'est pas résolu à ce jour dans la littérature.

Les matériels actuels utilisent des techniques corrélatives qui présentent une assez bonne robustesse sur de faibles intervalles de temps et se prêtent aisément à une implémentation en temps-réel vidéo. Néanmoins, ces techniques présentent des dérives sur des intervalles de temps plus ou moins longs lorsque la scène contient des éléments de reliefs (comme des bâtiments) importants ou de forts effets de perspective (en visée rasante par exemple).

Contributions

Nos contributions se placent dans le contexte de l'amélioration des algorithmes de poursuite existants, notamment en prenant en compte l'information 3D de la scène. Pour cela, nous nous plaçons dans un cadre bayésien, et formulons le suivi de manière probabiliste, au moyen d'un filtre particulaire. Nous avons mis en place trois algorithmes :

1. Le premier algorithme est basé sur des histogrammes de niveaux de gris, que l'on combine à un filtrage particulaire ; c'est un suivi purement 2D dans le sens où aucune information 3D réelle de la scène est utilisée.
2. Le second algorithme est basé sur des modèles géométriques (qui peuvent être 2D ou 3D), que l'on combine à un filtrage particulaire. Nous ajoutons une étape supplémentaire au filtrage particulaire classique, nous permettant de changer de modèle lorsque l'algorithme le juge nécessaire.
3. Le dernier algorithme combine les deux précédents ; l'intégration d'histogrammes de niveaux de gris et de comparaison de contours de modèles géométriques dans un filtrage

particulaire permet non seulement de rendre le suivi d'objets plus robuste, mais aussi de prendre en compte de l'information 3D réelle de la scène observée.

Plan de la thèse

Nous présentons au chapitre 1 les notions de base utilisées dans la thèse : les différents types de caméra, la géométrie épipolaire, l'estimation du mouvement de la caméra, des méthodes classiques d'optimisation utiles pour la compréhension de l'ensemble des travaux.

Le chapitre 2 traite du problème du suivi d'objets. Un état de l'art sur les différentes méthodes existantes met en évidence les enjeux et les difficultés de ces notions.

Le chapitre 3 présente la théorie du filtrage. Le filtrage linéaire, basé sur des hypothèses simples, permet de comprendre les bases. Nous décrivons en détail quelques filtres linéaires avant de présenter les filtres non linéaires, qui permettent de résoudre le problème du filtrage dans un cadre beaucoup plus général.

Le chapitre 4 présente nos contributions : trois algorithmes de suivi d'objet sont présentés. Nous nous plaçons dans un cadre probabiliste (le filtrage particulaire), ce qui nous permet d'avoir une grande flexibilité concernant le choix de la modélisation de l'ensemble des paramètres nécéssaires au suivi.

Les chapitres 5 et 6 analysent les performances des algorithmes développés et présentent l'ensemble des résultats obtenus. Le chapitre 5 se focalise sur la validation de l'algorithme de suivi d'objets basé sur des histogrammes de niveaux de gris. Le chapitre 6 quant à lui présente un protocole détaillé pour l'évaluation qualitative et quantitative de l'algorithme de suivi basé sur des modèles géométriques.

Enfin, les conclusions sur ce travail de thèse ainsi que les perspectives en termes d'amélioration des travaux existants ou d'applications supplémentaires sont présentées au chapitre 7.

Chapitre 1

Notions de base

Sommaire

1.1 Conventions et notations

Tout au long de cette thèse, les coordonnées homogènes sont utilisées pour décrire les entités géométriques : les points, droites et toute transformation de l'espace sont représentés par des vecteurs ou des matrices définis à un facteur multiplicatif non nul près. Cette relation d'égalité à un facteur près est notée $\sim$.

Les vecteurs sont représentés par des caractères gras, par exemple $\mathbf{q}$. La notation $\mathbf{q}$ représente un vecteur colonne ; un vecteur ligne est représenté par $\mathbf{q}^T$, où ^{T} désigne la transposition. Les matrices sont représentées par des majuscules, en caractères sans sérif, comme A. La transposée et l'inverse d'une matrice sont notées P^T et P^{-1}.

Les points 2D sont notés en minuscules $\mathbf{q}$, et les points 3D en majuscules $\mathbf{Q}$. De manière générale, les coordonnées d'un vecteur sont notées avec le même caractère, en italique, et un

indice indiquant sa position. Ainsi $\boldsymbol{Q}_i$ indique la $i^{\text{ème}}$ coordonnée du vecteur $\mathbf{Q}$.

Les matrices de projection sont notées P, ou $\mathsf{P_i}$ pour désigner la matrice de projection de l'image i.

Le produit scalaire de deux vecteurs $\mathbf{a}$ et $\mathbf{b}$ de longueur n est noté $\mathbf{a}^T\mathbf{b}$. Le symbole $\wedge$ est l'opérateur du produit vectoriel de deux vecteurs de longueur 3. Le vecteur $\mathbf{c} = \mathbf{a} \wedge \mathbf{b}$ est orthogonal à $\mathbf{a}$ et $\mathbf{b}$.

Dans cette thèse, nous travaillons avec différents espaces géométriques. L'espace Euclidien est celui qui nous entoure, dans lequel les transformations sont rigides et les grandeurs exprimées dans une unité de mesure. Par conséquent, nous utiliserons tout au long du manuscrit les distances suivantes :

- La distance point/point correspond à la longueur du segment reliant ces deux points. En dimension 3 et en utilisant les coordonnées homogènes, cela s'écrit :

$$dist(\mathbf{P}, \mathbf{Q}) = \| \frac{\mathbf{P}}{\mathbf{P}_4} - \frac{\mathbf{Q}}{\mathbf{Q}_4} \|^2$$

- La distance point/plan correspond à la distance entre le point et le projeté ortogonal de ce point sur le plan. La distance signée d'un point $\mathbf{Q}$ à un plan Π s'écrit :

$$dist(\mathbf{Q}, \Pi) = \frac{\Pi_1\mathbf{Q}_1 + \Pi_2\mathbf{Q}_2 + \Pi_3\mathbf{Q}_3 + \Pi_4\mathbf{Q}_4}{\mathbf{Q}_4\sqrt{\Pi_1^2 + \Pi_2^2 + \Pi_3^2}}$$

1.2 La décomposition en valeurs singulières

1.2.1 Définition

La décomposition en valeurs singulières d'une matrice (appelée « *Singular Value Decomposition* » en anglais, et couramment notée SVD) est un outil qui permet d'obtenir la solution d'un système linéraire.

Si A est une matrice de dimension (m,n), A peut être factorisée sous la forme :

$$\mathsf{A} = \mathsf{U}_{(\mathsf{m},\mathsf{n})}\mathsf{D}_{(\mathsf{m},\mathsf{n})}\mathsf{V}^{\mathsf{T}}_{(\mathsf{n},\mathsf{n})}$$

avec U matrice orthogonale de dimension (m,n) ($\mathsf{U}^{\mathsf{T}}\mathsf{U} = \mathsf{I}$), V^{T} matrice ortogonale de dimension (n,n) ($\mathsf{V}^{\mathsf{T}}\mathsf{V} = \mathsf{I}$) et D matrice diagonale dont les coefficients (tous positifs ou nuls) correspondent aux valeurs singulières de A. Le rang de A est égal au nombre de valeurs singulières non nulles.

1.2.2 Résolution d'un système linéaire au moindre carrés

Considérons la résolution aux moindres carrés d'un système linéaire et homogène

$$\min_X (\| \mathsf{A}\mathbf{x} \|_2)^2$$

Il faut éviter la solution triviale $\mathbf{x} = 0$. Cela se fait en imposant une contrainte sur la norme de la solution : $\| \mathbf{x} \|_2 = 1$ La solution de ce problème de minimisation contrainte est simple : il

suffit de décomposer en valeurs singulières la matrice A. Soit $\mathsf{A} = \mathsf{UDV}^\mathsf{T}$. La solution du système est alors donnée par :

$$\mathbf{x}^* = \mathbf{v}_n$$

où $\mathbf{v}_n$ est la dernière colonne de de V. Notons que la contrainte $\| \mathbf{x} \|_2 = 1$ est satisfaite par l'orthogonalité de la matrice V.

1.2.3 Applications

Les applications en vision par ordinateur utilisant cette technique sont nombreuses ; parmi les plus classiques, on trouve :
- le calibrage
- la détermination de la matrice fondamentale à partir de correspondances de points
- l'estimation du mouvement

1.3 Méthodes d'estimation/optimisation robustes

Ce paragraphe a pour objectif de présenter un rapide état de l'art des méthodes robustes d'estimation et d'optimisation utilisées en vision par ordinateur. Les problèmes classiques en vision par ordinateur qui requièrent l'utilisation de ce type de méthodes sont le calcul de pose, le suivi dans une séquence, l'ajustement de faisceaux pour recontruire une scène 3D...

De manière générale, on dit qu'une méthode est *robuste* si elle conserve ses propriétés malgré les incertitudes sur le modèle, les erreurs de mesures ou tout autre changement de l'environnement. Les caméras fournissent des informations bas niveau d'une grande richesse de la scène observée, mais en pratique il est très souvent difficile d'en extraire l'information haut niveau pertinente pour l'application visée. A partir des informations fournies par une caméra et du processus de formation des images (modèle de caméra), il est possible d'estimer les paramètres recherchés. Ces paramètres peuvent être, selon les applications en vision par ordinateur, une information de profondeur (une information 3D sur la scène), la position de la caméra par rapport à la scène (encore appelée pose de la caméra), le mouvement de la caméra ou d'objets dans la scène, etc.

Des modèles mathématiques (comme la géométrie projective) permettent en théorie de fournir un cadre adapté à la modélisation de la géométrie de l'environnement ou du processus d'acquisition des caméras. Cependant, dès que l'on se confronte à des images réelles, la modélisation devient souvent inexacte, et des algorithmes robustes sont souvent nécessaires. D'autre part, des changements d'illumination de la scène ou encore des occultations sont souvent à l'origine de mesures abbérantes, que l'on appelera par la suite les erreurs grossières (en anglais, les « *outliers* »).

Pour résoudre le problème posé compte tenu des erreurs grossières et de l'inexactitude du modèle théorique, on trouve deux types d'approches. Le premier permet de résoudre simultanément le problème de la détection des outliers et de l'estimation. Les algorithmes utilisés classiquement sont les suivants : LMedS (Least Median Square) et LTS (Least Trimmed Squares) [RL87], les M-estimateurs, L-estimateurs ou R-estimateurs [Hub81]. Le second type d'approche consiste à détecter les outliers, avant de procéder à l'estimation de paramètres. Parmi ces approches on

trouve la transformée de Hough [Hou62] et l'algorithme RANSAC (Random Sample Consensus) [FB81].

Les M- R- et L-estimateurs

Ce sont des estimateurs proposés dans la statistique robuste pour compenser des erreurs grossières. Le principe des L-estimateurs est d'omettre une proportion α des plus petites valeurs, et une autre proportion α dse plus grandes, puis de calculer la moyene de l'ensemble des valeurs restantes.

Les M-estimateurs minimisent des fonctions objectives plus générales que l'habituelle somme des carrés des résidus associée à la moyenne d'un échantillon. Plutôt que de prendre le carré de l'écart entre chaque observation $\mathbf{Y}$ et l'estimateur e, on utilise une fonction $\rho(\mathbf{Y}, e)$ et former la fonction objective en faisant la somme sur tout l'échantillon $\sum_{i=1}^{n} \rho(Y_i, e)$. Souvent, la fonction $\rho(\mathbf{Y}, e)$ ne dépend que de $\mathbf{Y}$ et de e qu'à travers $\mathbf{Y} - e$.

Enfin, les R-estimateurs ne s'appuie plus sur des relations linéaires mais tient compte du classement des écarts. Pour plus de détails sur ces estimateurs robustes, on peut se référer au document [Jol06].

La transformée de Hough

La transformée de Hough est une méthode de vote très robuste. Elle repose sur la discrétisation de l'espace des paramètres, et à chaque hypercube (de l'espace d'état) est associé un acccumulateur. L'algorithme fonctionne ensuite de la manière suivante : pour un jeu de données de taille minimale, on estime les paramètres recherchés et on incrémente l'accumulateur de l'hypercube correspondant. Puis on recommence avec un autre jeu de données jusqu'à considérer l'ensemble des combinaisons possibles des données disponibles. L'accumulateur qui a recu le plus de vote correspond alors à la meilleure estimation des paramètres.

Cette méthode est tout à fait adaptée aux problèmes qui ont un grand nombre de données par rapport au nombre de paramètres à estimer. Si les données ne sont pas en nombre suffisant, aucun accumulateur n'est prépondérant. Ceci dit, cette transformée est très robuste puisqu'elle effectue une recherche exhaustive et globale sur l'ensemble des données.

L'algorithme RANSAC

La méthode RANSAC (en anglais « Random Sample Concensus ») est également une procédure de vote, mais à la différence de la méthode précédente, c'est ue méthode probabiliste. Elle a été proposée pour réduire les temps de calcul des procédures de votes classiques. L'algorithme consiste à estimer les paramètres recherchés avec le minimum de données nécéssaires. Ces donnés nécéssaires sont tirées aléatoirement dans l'ensemble des données. On compte ensuite le nombre de mesures compatibles avec le modèle déduit : c'est l'ensemble des consensus. On réitère ces étapes de tirage au sort et de dénombrement des mesures compatibles, et on retient finalement le modèle qui maximise l'ensemble des consensus.

Il y a quelques paramètres à regler pour le bon fontionnement de l'algorithme : tout d'abord, il faut connaître le nombre de tirages au sort à effectuer pour être sur (avec une certaine probabilité) de trouver le meilleur ensemble de paramètres. En appelant w la probablité que nimporte quelle

mesure soit compatible avec le modèle cherché, k le nombre de tirages à réaliser, m le nombre de mesures nécéssaires pour estimer les paramètres cherchés et p la probabilité qu'au moins un ensemble de mesures soit sans erreur, on a alors $(1 - w^m)$ la probabilité qu'aucun ensemble de m mesures soit compatible avec les modèle. On en déduis alors que

$$(1 - w^m)^k = 1 - p$$

Finalement, le nombre de tirages au sort à effectuer est :

$$k = \frac{log(1-p)}{log(1-w^m)}$$

Cet algorithme est robuste aux erreurs grossières, et a été comparé à d'autres estimateurs. On peut se référer à travaux de Meer [MMKR91] et aux travaux de Lan [Lan97] pour plus de détails sur les différents estimateurs robustes.

1.4 Modèles de caméra

La caméra est l'outil qui fait le lien entre le monde 3D et son image 2D. L'utilisation de la géométrie projective permet d'écrire de manière simple, sous une forme algébrique, cette relation qui existe entre ces deux espaces. Ce paragraphe a pour but de présenter les outils existant en vision par ordinateur pour extraire les informations utiles dans une image. A partir d'un modèle de caméra simple, il est possible de remonter à la reconstruction 3D de la scène grâce à la géométrie projective : nous détaillerons ces étapes.

Les modèles de caméra sont des opérateurs géométriques basés sur le principe du modèle sténopé. Selon les hypothèses qu'il est possible de faire sur ce modèle on distingue différents types de caméra : la caméra perspective et la caméra affine.

1.4.1 Le modèle perspectif (ou sténopé)

Sur le plan géométrique, on peut décrire ce modèle par un centre de projection (encore appelé *centre optique*) et un plan image. Un point 3D de la scène est projeté le long du rayon qui le lie avec le centre de projection ; l'intersection de ce rayon (encore appelé *rayon de projection* ou *ligne de vue*) avec le plan image est appelée point image.

La droite qui passe par le centre de projection et qui est perpendiculaire au plan image est appelée axe optique. Le point d'intersection de l'axe optique et du plan image est le point principal. Toutes ces définitions et notations sont représentées sur la figure 1.1.

La matrice de projection P représente la transformation projective du monde 3D vers le monde 2D. L'équation de projection est donnée par :

$$\mathbf{q} \sim \mathsf{P}\mathbf{Q}$$

On peut, en décomposant la matrice P, faire apparaitre le lien entre la matrice de projection et les grandeurs physiques qui caractérisent la caméra. Cette décomposition se fait en paramètres intrinsèques (encore appelés paramètres internes) et paramètres extrinsèques (ou paramètres externes). Les paramètres intrinsèques représentent les caractéristiques internes de la caméra, qui

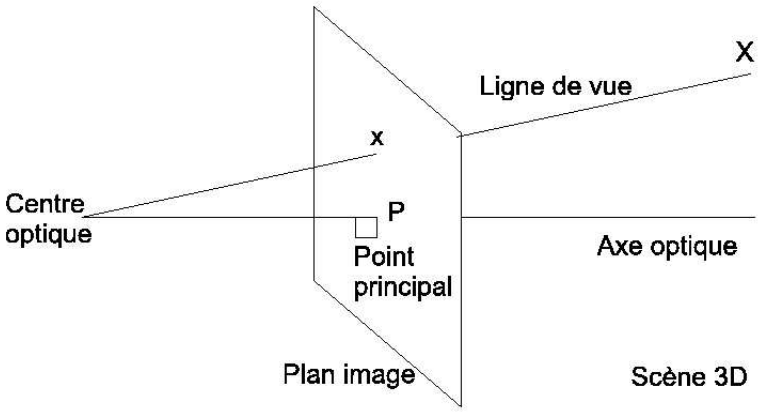

FIG. 1.1 – Le modèle de caméra sténopé

sont invariantes à sa position. Ce sont la distance focale (exprimée en pixels) et les coordonnées du point principal. Les paramètres extrinsèques représentent la position et l'orientation de la caméra par rapport au monde réel.

La décomposition de P conduit à l'équation suivante :

$$\mathsf{P} \sim \mathsf{K}\,(\mathsf{R}\,\mathbf{t})$$

Dans cette équation, K est une matrice triangulaire supérieure qui modélise les paramètres internes de la caméra, et R et $\mathbf{t}$ représentent respectivement la rotation et la translation de la caméra par rapport au monde réel (encore appelé la *pose* de la caméra).

1.4.2 Le modèle affine

Une caméra affine est une caméra dont la matrice de projection est une transformation affine, c'est-à-dire de la forme suivante :

$$\mathsf{P} \sim \begin{pmatrix} \mathsf{P}_{2\times 3} & \mathbf{p}_2 \\ \mathbf{0}_3^T & 1 \end{pmatrix}$$

Une condition nécéssaire et suffisante pour qu'une caméra soit affine est que son centre optique soit à l'infini. Avec ce modèle, les concepts d'axe optique et de point principal n'ont plus aucun sens. Toutes les droites sont parallèles et perpendiculaires au plan image. Notons que dans le cas des caméras orthographiques, qui sont une sous-famille des caméras affines, toutes les droites perpendiculaires au plan image passent par le centre de projection.

Le fonctionnement d'une caméra affine est équivalent à une projection parallèle à un facteur d'échelle près. Si on considère que le monde est localement plat [Fau85], ce modèle est une approximation utile dans de nombreux cas pratiques. Comme toute transformation affine, la caméra affine préserve le parallélisme.

1.5 La géométrie épipolaire

On parle de géométrie épipolaire lorsque l'on considère deux images prises avec une caméra sous deux points de vue différents. Considérons $\mathbf{M}$ un point de la scène 3D considérée, et $\mathbf{m}_1$ et $\mathbf{m}_2$ les projections de ce point $\mathbf{M}$ dans les deux images. On notera $\mathbf{C}_1$ et $\mathbf{C}_2$ les centres de projection de la caméra pour chacune des images. Par géométrie, on sait que la position du point $\mathbf{M}$ se trouve à l'intersection des rayons optiques le liant respectivement à $\mathbf{m}_1$ et à $\mathbf{m}_2$. Toutes ces notations sont reprises sur la figure 1.2. Théoriquement, il est donc possible de reconstituer, à partir des seules images en 2D, la réalité 3D. L'ensemble formé par les trois points $\mathbf{M}$, $\mathbf{m}_1$ et $\mathbf{m}_2$ et par les droites qui les relient définit ce que l'on appelle la *contrainte épipolaire*. L'objectif de la géométrie épipolaire est non seulement de reconstruire la troisième dimension à partir des images, mais surtout d'effectuer l'appariement des points dans chacune des deux images.

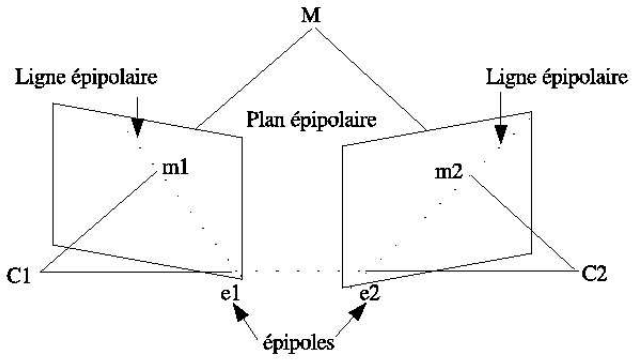

FIG. 1.2 – La géométrie épipolaire

Le plan passant par le point $\mathbf{M}$ de la scène et ses deux projections $\mathbf{m}_1$ et $\mathbf{m}_2$ est appelé *plan épipolaire*. Ce plan coupe les deux plans image en deux droites, les *droites épipolaires*, que l'on note respectivement $\mathbf{l}_1$ et $\mathbf{l}_2$. La configuration est parfaitement symétrique : le point dans la première image qui correspond à un point $\mathbf{m}_2$ dans la deuxième image se trouve sur la droite épipolaire associée $\mathbf{l}_1$.

La géométrie épipolaire facilite la recherche de correspondants d'une image à l'autre. En effet, considérons le cas où on cherche le correspondant d'un point $\mathbf{m}'_1$ qui se trouve sur la droite épipolaire $\mathbf{l}_1$ associée à $\mathbf{m}_1$. On peut observer que le plan épipolaire, ainsi que les deux droites épipolaires, sont les mêmes que ceux associés à $\mathbf{m}_1$. On peut en conclure que toutes les paires formées d'un point sur $\mathbf{l}_1$ et d'un autre sur $\mathbf{l}_2$ sont des correspondances possibles.

Lorsque le point $\mathbf{m}'_1$ ne se trouve plus sur la droite épipolaire $\mathbf{l}_1$, cela donne lieu à un nouveau plan épipolaire. Par construction, ces deux plans épipolaires contiennent la ligne qui joint les deux centres de projection : la *ligne de base*. En répétant cette construction un grand nombre de fois, on produit un faisceau de plans épipolaires, auquel correspondent les deux faisceaux de droites épipolaires. Les bases de ces deux faisceaux de droites épipolaires sont deux points, les épipoles, notés $\mathbf{e}_1$ et $\mathbf{e}_2$.

Remarquons qu'on peut voir l'épipole $\mathbf{e}_1$ comme l'image du centre de projection de la deuxième caméra $\mathbf{C}_2$; et réciproquement $\mathbf{e}_2$ est l'image de $\mathbf{C}_1$.

1.5.1 La matrice fondamentale

La matrice fondamentale est un concept clé pour toutes les questions touchant à l'emploi d'images non calibrées prises de points de vue différents. Elle contient toute l'information géométrique disponible. Elle représente lagéométrie épipolaire de deux vues non calibrées. Elle constitue une reconstruction implicite des deux caméras. Cela se traduit par des contraintes sur des correspondances de points, c'est-à-dire des points images provenant d'un même point 3D. Ainsi, si $\mathbf{m}_1$ et $\mathbf{m}_2$ sont les projections d'un même point 3D $\mathbf{M}$, la matrice fondamentale, que l'on note F, satisfait la relation :

$$\mathbf{m}_2^T \mathsf{F} \mathbf{m}_1 = 0$$

La matrice fondamentale est représentée par une matrice 3x3 de rang 2 ; elle est définie à un facteur d'échelle et ne dépend que des paramètres intrinsèques et des paramètres extrinsèques de la caméra.

Dans la littérature, on trouve différentes méthodes pour le calcul de la matrice fondamentale. Parmi elles, l'algorithme de Longuet-Higgins (souvent appelé algorithme des 8 points) amélioré par Hartley en 1995, des algorithmes robuste (de type RANSAC par exemple) ou encore des méthodes non linéaires.

Méthode linéaire : la méthode des 8 points

Longuet-Higgins [LH81] propose de reconstruire la géométrie épipolaire en estimant la matrice fondamentale à partir de 8 correspondances de points et sans information sur les paramètres intrinsèques ou extrinsèques de la caméra.

Chaque correspondance de point fournit une équation linéaire homogène sur F :

$$\mathbf{m}_2^T \mathsf{F} \mathbf{m}_1 = 0$$

Les coordonnées utilisées ici sont des coordonnées homogènes ; on a donc $m_{13} = m_{23} = 1$. Cette équation peut s'écrire sous une forme linéaire de la manière suivante :

$$\mathbf{a}^T \mathbf{X} = 0$$

$$\text{où} \quad \mathbf{a} = (\mathbf{m}_{11}\mathbf{m}_{21}, \mathbf{m}_{11}\mathbf{m}_{22}, \mathbf{m}_{11}, \mathbf{m}_{12}\mathbf{m}_{21}, \mathbf{m}_{12}\mathbf{m}_{22}, \mathbf{m}_{12}, \mathbf{m}_{21}, \mathbf{m}_{22}, 1)$$

$$\text{et} \quad \mathbf{X} = (F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33})$$

Cette équation est valable pour chaque paire de points en correspondance. Ainsi pour n couples de points, on obtient le système linéaire suivant :

$$\mathsf{A}\mathbf{X} = 0 \tag{1.1}$$

où A est une matrice $n \times 9$, et $\mathbf{X}$ est le vecteur dont les composantes sont les 9 éléments de la matrice fondamentale.

Etant donné l'ensemble des correspondances de points :

1. Construire le système homogène $\mathsf{A}\mathbf{X} = 0$ où A est une matrice $n \times 9$, et résoudre le système linéaire en utilisant la SVD (voir 1.2 pour plus de détails) ; on a donc $\mathsf{A} = \mathsf{U}\mathsf{D}\mathsf{V}^\mathsf{T}$
2. Affecter 0 à la valeur singulière la plus petite : soit D_c la matrice ainsi corrigée
3. La matrice F corrigée est alors égale à $\mathsf{F}_\mathsf{c} = \mathsf{U}\mathsf{D}_\mathsf{c}\mathsf{V}^\mathsf{T}$ ce qui est la matrice singulière la plus proche de F au sens de la norme de Frobenius

FIG. 1.3 – Algorithme des 8 points.

A cause du bruit dans les données, il n'y a en général pas de solution exacte pour l'équation (1.1) ; on résoud donc le système aux moindres carrés, c'est-à-dire que l'on cherche $\hat{\mathbf{X}}$ tel que :

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{\operatorname{argmin}} \|\mathsf{A}\mathbf{X}\|^2 \quad \text{sous la contrainte} \quad \|\mathbf{X}\| = 1$$

En utilisant la contrainte que le rang de F doit être égal à 2, on détermine la solution au problème ; ce sont les étapes 3,4 et 5 de l'algorithme décrit sur la figure 1.3.

Le système (1.1) est un système d'équation linéaire et homogène. De manière générale, s'il n'y a pas de bruit sur les données, en présence de m inconnues, il y a une solution unique (à l'échelle près) si l'on dispose de $m-1$ équations. Dans notre cas, chaque correspondance de points fournit une équation : il suffit donc de 8 correspondances de points pour estimer la géométrie épipolaire. C'est d'ailleurs pour cela que l'algorithme est aussi appelé *méthode des 8 points* dans la littérature. Il est résumé sur la figure 1.3.

L'encapsulation de ce dernier algorithme en utilisant la méthode RANSAC permet de rechercher une solution avec le minimum de correspondances de points. Les autres correspondances de points disponibles permettent ensuite de confirmer cette première estimation. Si un consensus est obtenu, l'estimation est retenue. Pour plus de détails sur ce type de méthode, on peut se référer à la section 1.3 (page 4).

L'intérêt d'utiliser RANSAC pour le calcul de la matrice fondamentale est que l'algorithme est robuste aux erreurs d'appariement, contrairement à l'algorithme proposé ci dessus. Hartley [Har97] a proposé un algorithme en normalisant les points avant l'estimation robuste de F.

Méthode non linéaire

On peut également estimer la matrice fondamentale par des méthodes non linéaires ; par exemple, une solution consiste à minimiser la somme des carrés des distances d'un point à la droite épipolaire qui est censée passer par ce point. En effet l'équation $\mathbf{m}_1^T F \mathbf{m}_2 = 0$ exprime que le point $\mathbf{m}_2$ se trouve sur la droite épipolaire $\mathbf{l}_2$ dont les paramètres sont donnés par l'expression :

$$\begin{pmatrix} \mathbf{l}_{21} \\ \mathbf{l}_{23} \\ \mathbf{l}_{23} \end{pmatrix} = \begin{pmatrix} \mathsf{F}_{11} & \mathsf{F}_{12} & \mathsf{F}_{13} \\ \mathsf{F}_{21} & \mathsf{F}_{22} & \mathsf{F}_{23} \\ \mathsf{F}_{31} & \mathsf{F}_{32} & \mathsf{F}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{m}_{11} \\ \mathbf{m}_{12} \\ 1 \end{pmatrix}$$

ou, qui s'écrit plus simplement

$$\mathbf{l}_2 = \mathsf{F}\mathbf{m}_1$$

La distance d'un point $\mathbf{m}_2$ à cette droite est donnée par l'expression suivante :

$$d(\mathbf{m}_2, \mathbf{l}_2) = \frac{\mid \mathbf{m}_2^T \mathbf{l}_2 \mid}{\sqrt{\mathbf{l}_{21}^2 + \mathbf{l}_{22}^2}}$$

On veut minimiser la somme des carrés de ces distances d. En pratique, Luong et Faugeras [LF96] ont proposé de considérer simultanément la distance de $\mathbf{m}_2$ à la droite $\mathsf{F}\mathbf{m}_1$ et la distance de $\mathbf{m}_1$ à la droite $\mathsf{F}^\mathsf{T}\mathbf{m}_2$ afin de réduire les différences entre la géométrie épipolaire gauche-droite et la géométrie epipolaire droite-gauche. Pour n correspondances de points, il faut minimiser le critère suivant :

$$min_\mathsf{F} \left[\sum_i \left(\frac{(\mathbf{m}_2^T \mathsf{F}\mathbf{m}_1)^2}{(\mathsf{F}\mathbf{m}_1)_1^2 + (\mathsf{F}\mathbf{m}_1)_2^2} \right) + \sum_i \left(\frac{(\mathbf{m}_2^T \mathsf{F}\mathbf{m}_1)^2}{(\mathsf{F}^\mathsf{T}\mathbf{m}_2)_1^2 + (\mathsf{F}^\mathsf{T}\mathbf{m}_2)_2^2} \right) \right]$$

Cette minimisation implique l'utilisation de techniques d'optimisation non linéaires telles que Newton, Gauss-Newton ou Levenberg-Marquardt. On peut se référer au document [PFTV93] pour les détails de ces méthodes.

1.5.2 La matrice essentielle

Comme on vient de l'expliquer, la matrice F est estimée en utilisant les coordonnées pixels des images ; elle dépend des paramètres intrinsèques et extrinsèques de la caméra. Lorsque la caméra est calibrée et que l'on utilise les coordonnées images normalisées, la géométrie épipolaire est décrite par la matrice essentielle E [HM93].

Cette matrice est définie comme une fonction du mouvement existant entre les deux caméras ; en notant R la matrice de rotation entre les deux caméras, et $\mathbf{t}$ la translation, la matrice essentielle peut s'écrire :

$$\mathsf{E} = [\mathbf{t}]_\times \mathsf{R} = \mathsf{R}[\mathsf{R}^\mathsf{T}\mathbf{t}]_\times$$

En utilisant la matrice essentielle, on peut montrer que l'équation de la droite épipolaire qui, à un point de l'image gauche $\mathbf{m}_1$ fait correspondre une droite de l'image de droite $\mathbf{m}_2$, s'écrit :

$$\mathbf{m}_2^T \mathsf{E}\mathbf{m}_1 = 0$$

On peut en déduire une autre expression de E, très utilisée en pratique :

$$\mathsf{E} = \mathsf{K}^\mathsf{T}\mathsf{F}\mathsf{K}$$

On peut se reporter au livre de Hartley et Zisserman [HZ00] pour les détails et les démonstrations.

1.5.3 Cas de scènes planes : homographie

Deux images provenant de la vue d'une scène plane depuis des points de vue différents sont liés par une transformation appelée homographie. Si $\mathbf{x}_1$ est un point de la première image et $\mathbf{x}_2$ son correspondant dans la deuxième image, la relation liant ces points à l'homographie s'écrit :

$$\mathbf{x}_1 \sim \mathsf{H}_{12}\mathbf{x}_2$$

Si la scène est plane, on peut montrer que les projections $\mathbf{x}_1$ et $\mathbf{x}_2$ d'un point $\mathbf{X}$ situé sur le plan sont reliés par une homographie qui dépend uniquement des positions relatives des deux centres de projection. De plus, cette homographie est reliée aux paramètres géométriques par la relation suivante :

$$\mathsf{H} = \mathsf{dR} + \mathbf{tn}$$

où $\mathbf{n}$ est le vecteur normal au plan, d est la distance à l'origine du système de coordonnées de la première caméra et $\mathbf{t}$ est la translation entre les deux centres de projection des caméras. Pour la démonstration de ce résultat, on peut se référer aux travaux de Faugeras [FL88].

1.6 L'estimation du mouvement de la caméra

De nombreux travaux traitent de l'estimation du mouvement dans une séquence d'images. L'expression « estimation du mouvement » est ambiguë, et parfois mal employée. Il faut distinguer l'estimation du mouvement apparent, c'est-à-dire le mouvement des objets dans les images, du mouvement réel des objets dans la scène 3D et/ou de celui de la caméra dans la scène réelle.

Nous nous plaçons dans le cas où on cherche à estimer le mouvement d'une caméra mobile dans une scène 3D. Plusieurs méthodes existent et nous allons les détailler dans les prochains paragraphes.

1.6.1 Les méthodes classiques

Il est possible, à partir de la matrice fondamentale, de reconstruire la 3D à une homographie près. Si on connaît le calibrage de la caméra, on peut reconstruire la scène 3D à une similitude de l'espace près.

Considérons deux images $\mathbf{m}_1$ et $\mathbf{m}_2$ et supposons la matrice essentielle connnue. D'autre part, supposons que la matrice de projection de la première image s'écrit $\mathsf{P}_1 = \mathsf{K}(\mathsf{I} \mid 0)$. Pour calculer P_2, il est nécessaire de décomposer E en valeur singulière. Cette décomposition permet de remonter ensuite à la translation, puis à la rotation de la matrice P_2. Il y a quatres solutions. Si $\mathsf{E} = \mathsf{U}\mathrm{diag}(1,1,0)\mathsf{V}^\mathsf{T}$, ces solutions s'écrivent [HZ00] :

$$\mathsf{P}_2 = [\mathsf{UWV}^\mathsf{T} \mid \mathsf{U}_3] \qquad \mathsf{P}_2 = [\mathsf{UWV}^\mathsf{T} \mid -\mathsf{U}_3] \qquad \mathsf{P}_2 = [\mathsf{UW}^\mathsf{T}\mathsf{V}^\mathsf{T} \mid \mathsf{U}_3] \qquad \mathsf{P}_2 = [\mathsf{UW}^\mathsf{T}\mathsf{V}^\mathsf{T} \mid -\mathsf{U}_3]$$

où W est la matrice $\mathsf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

La différence entre les deux premières solutions est la direction du vecteur translation qui est inversée. Les solutions 1 et 3 correspondent à des solutions « twistées ». Parmi les quatre

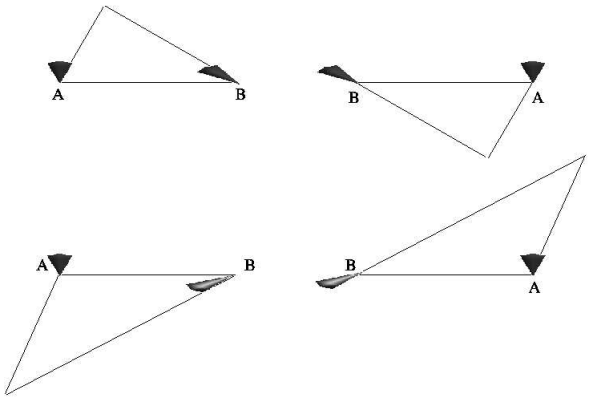

FIG. 1.4 – Les 4 solutions pour la reconstruction à partir de la matrice essentielle E : chaque dessin représente un point 3D et les deux lignes de vues. Entre les colonnes droites et gauche, il y a une symétrie par rapport à l'axe AB ; entre les lignes du haut et du bas, la caméra B tourne de 180 degrés par rapport à la ligne de base AB. Des 4 configurations, une seule permet à un point 3D d'être devant les deux caméras.

solutions, une seule permet en général à un point 3D de la scène d'être vu par les deux caméras. Il faut donc tester chacune des configurations pour déterminer la bonne solution.

Le schéma général pour estimer le mouvement de la caméra à l'aide de la géométrie épipolaire est le suivant : La première étape est celle de la mise en correspondance des images avec une méthode non contrainte par la géométrie épipolaire (KLT par exemple). En général, le résultat de cette étape est infesté d'erreurs grossières. A partir de cette mise en correspondance, on estime de manière robuste la géométrie épipolaire. L'estimation robuste (de type RANSAC) permet de gérer les erreurs de bruits et les erreurs grossières de manière efficace. A ce point, on a donc estimé la matrice fondamentale F. Si on veut une estimation plus précise, on peut effectuer une mise en correspondance plus fine, c'est-à-dire sur des points image non pertinents, en utilisant la géométrie épipolaire. Puis on réestime la géométrie épipolaire avec l'ensemble des correspondances de points obtenues. Puis, à partir de F estimée et K la matrice de calibration connue, on calcule la matrice essentielle E en utilisant la relation $\mathsf{E} = \mathsf{K}^T\mathsf{F}\mathsf{K}$. Il ne reste plus qu'à extraire le mouvement ; pour cela, on teste chacune des 4 solutions précédentes pour ne conserver que la bonne. L'algorithme est résumé sur la figure 1.5 ci-dessous.

1.6.2 Une méthode pour des scènes planes

Lorsque l'on se place dans le cas où les images acquises sont des images aériennes, le rapport entre la taille des objets 3D de la scène et la distance de la caméra à la scène est très faible. L'algorithme général et décrit 1.5 ne peut plus s'appliquer. Il faut donc trouver un autre moyen pour estimer le mouvement de la caméra à partir des données images.

Nous faisons l'hypothèse suivante : puisque l'on considère des images aériennes, on est loin des objets et a priori il devrait y avoir un plan dominant dans les images. Ce plan dominant peut être le plan du sol, ou des toits si l'avion survole une zone urbaine. Cette hypothèse s'avère réaliste dans les cas que nous avons traités.

1. Mise en correspondance des images sans contrainte épipolaire (KLT)
2. Estimation robuste (RANSAC) de la géométrie épipolaire
3. (Optionnelle) Mise en correspondance plus fine
4. Calcul de E à partir de F et K : $\mathsf{E} = \mathsf{K}^\mathsf{T}\mathsf{F}\mathsf{K}$
5. Extraction du mouvement à partir de E

FIG. 1.5 – Un schéma général pour l'estimation du mouvement à partir de la géométrie épipolaire

On suppose donc que la calibration intrinsèque de la caméra est connue et que l'on a un plan dominant dans la scène, le plus souvent ce plan correspond au plan du sol. A partir de correspondances de points dans les images, on estime de manière robuste l'homographie. La mise en correspondance des points se fait de la même manière que précédemment (en utilisant KLT), et l'homographie est calculée avec un algorithme robuste (RANSAC). L'homographie calculée, que l'on notera par la suite H, peut également s'écrire comme une fonction des paramètres du mouvement ; elle s'écrit comme la somme d'une matrice de rotation et d'une matrice de rang 1 [FL88] :

$$\mathsf{H} = \mathsf{R} + \frac{\mathbf{t}}{\mathsf{d}}\mathbf{n}^\mathsf{T} \tag{1.2}$$

où $\mathbf{t}$ est la translation de la caméra entre les deux vues, d la distance entre le premier centre de projection et le plan dominant, et $\mathbf{n}$ le vecteur normal au plan.

L'équation (1.2) a en général 9 solutions différentes. Si H a une valeur singulière de multiplicité double, il n'y a plus que 4 solutions au problème. Enfin, le problème est partiellement indéterminé si une des valeurs singulières de H est de multiplicité 3.

Pour trouver les solutions au problème, on fait une décomposition en valeurs singulières de H : on peut toujours décomposer H sous la forme $\mathsf{H} = \mathsf{U}\mathsf{D}\mathsf{V}^\mathsf{T}$. On classe les valeurs singulières d_i de la matrice D par ordre décroissant, on a donc $d_1 \geq d_2 \geq d_3$.

En utilisant cette décomposition, on obtient une nouvelle équation :

$$\mathsf{D} = \mathsf{R}' + \frac{\mathbf{t}'}{\mathsf{d}'}\mathbf{n}'^\mathsf{T} \tag{1.3}$$

Notons que R, $\mathbf{t}$ et $\mathbf{n}$ sont reliés à R', $\mathbf{t}'$ et $\mathbf{n}'$ par les relations

$$\begin{cases} \mathsf{R} &= s\mathsf{U}\mathsf{R}'\mathsf{V}^\mathsf{T} \\ \mathbf{t} &= \mathsf{U}\mathbf{t}' \\ \mathbf{n} &= \mathsf{V}\mathbf{n}' \\ d &= sd' \\ s &= det\mathsf{U}det\mathsf{V} \end{cases}$$

On remarque que R' est une rotation : son déterminant vaut 1, et le produit de R' et de sa transposée vaut l'identité I. D'autre part, en utilisant la base canonique, et en écrivant $\mathbf{n}'$ sur cette base, on obtient la relation :

$$\mathbf{n}' = \mathbf{n}_1\mathbf{e}_1 + \mathbf{n}_2\mathbf{e}_2 + \mathbf{n}_3\mathbf{e}_3$$

En utilisant ces relations, l'équation (1.3) nous fournit 3 équations ; pour i valant 1,2 ou 3, on a l'équation :

$$d_i\mathbf{e}_i = d'R'\mathbf{e}_i + \mathbf{t}'\mathbf{n}_i$$

On impose que la norme de $\mathbf{n}$ vaut 1, et que V est orthogonal ; $\mathbf{n}'$ est donc aussi de norme 1, ce qui s'exprime par la relation $\sum_{i=1}^{3}\mathbf{n}_i^2 = 1$. En éliminant $\mathbf{t}'$ on obtient finalement :

$$d'\mathsf{R}'(\mathbf{n}_\mathsf{j}\mathbf{e}_\mathsf{i} - \mathbf{n}_\mathsf{i}\mathbf{e}_\mathsf{j}) = \mathsf{d}_\mathsf{i}\mathbf{n}_\mathsf{j}\mathbf{e}_\mathsf{i} - \mathsf{d}_\mathsf{j}\mathbf{n}_\mathsf{i}\mathbf{e}_\mathsf{j} \qquad \text{pour} \quad \mathsf{i} \neq \mathsf{j}$$

Comme R' préserve la norme des vecteurs, on en tire 3 équations

$$\begin{cases} (d'^2 - d_2^2)\mathbf{n}_1^2 + (d'^2 - d_1^2)\mathbf{n}_2^2 &= 0 \\ (d'^2 - d_3^2)\mathbf{n}_2^2 + (d'^2 - d_2^2)\mathbf{n}_3^2 &= 0 \\ (d'^2 - d_1^2)\mathbf{n}_3^2 + (d'^2 - d_3^2)\mathbf{n}_1^2 &= 0 \end{cases}$$

On peut voir ce système d'équations comme un système linéaire en les inconnues $\mathbf{n}_1$, $\mathbf{n}_2$ et $\mathbf{n}_3$. Comme la solution ne peut pas être nulle, le déterminant doit être nul, ce qui s'écrit :

$$(d'^2 - d_1^2)(d'^2 - d_2^2)(d'^2 - d_3^2) = 0$$

Il faut considérer différents cas, selon l'ordre de multiplicité des valeurs singulières de D :

1. $d_1 \neq d_2 \neq d_3$ et donc $d' = \pm d_2$
2. $d_1 = d_2 \neq d_3$ ou $d_1 \neq d_2 = d_{13}$, et $d' = \pm d_2$
3. $d_1 = d_2 = d_3$ et $d' = \pm d_2$

Il faut également distinguer pour l'étude les cas où d' est positif ou négatif. La synthèse des différents cas est présentée sur la figure 1.6.

En pratique, on n'a aucun moyen de connaître le signe de d' a priori. On peut donc conclure sur le nombre de solutions au problème de la manière suivante : il y a 8 solutions lorsque les valeurs singulières sont distinctes, 4 solutions lorsque deux valeurs singulières sont égales, et une indétermination lorsque les trois sont égales. Mais seules deux solutions parmi les 8 sont valides en terme d'interprétation physique des résultats.

On peut se reporter aux travaux de Faugeras et Lustman [FL88] pour les détails et les démonstrations de ces résultats.

Comme on vient de le voir, souvent deux solutions satisfont au problème. Comment alors trouver la bonne solution ? Si l'on dispose d'informations supplémentaires, la normale au plan par exemple, l'une des deux solutions peut être éliminée. Sinon, on peut envisager les solutions suivantes :

- Utiliser une troisième image
- Si un deuxième plan est présent dans l'image, on peut procéder de la même manière, et choisir la solution commune aux deux paires.

- Utiliser des relations géométriques des images que l'on connaît (deux droites orthogonales par exemple)
- Tester chacune des deux solutions pour ne conserver que la bonne !

L'algorithme d'estimation des paramètres de mouvement à partir d'un plan de référence est résumé sur la figure 1.6.

1. Mise en correspondance des images sans contrainte épipolaire (KLT)
2. Estimation robuste (RANSAC) de l'homographie H
3. Décomposition en valeurs singulières de H sous la forme $\mathsf{H} = \mathsf{U}\mathsf{D}\mathsf{V}^{\mathsf{T}}$
4. En fonction des valeurs singulières de D
   - si $d' \geq 0$
     - si $d_1 \neq d_2 \neq d_3$ et donc $d' = d_2$: on a une solution R' et $\mathbf{t}'$
     - si $d_1 = d_2 \neq d_3$ ou $d_1 \neq d_2 = d_{13}$, et $d' = d_2$

$$\begin{cases} R' &= I \\ \mathbf{t}' &= (d_1 - d_3) \begin{pmatrix} \mathbf{n}_1 \\ 0 \\ -\mathbf{n}_3 \end{pmatrix} \end{cases}$$

     - si $d_1 = d_2 = d_3$ et $d' = d_2$
       on a une rotation pure ; la normale n'est pas définie

$$\begin{cases} R' &= I \\ \mathbf{t}' &= 0 \end{cases}$$

   - si $d' \leq 0$
     - si $d_1 \neq d_2 \neq d_3$ et donc $d' = -d_2$

$$\begin{cases} R' & \text{est une symmétrie} \\ \mathbf{t}' &= (d_1 + d_3) \begin{pmatrix} n_1 \\ 0 \\ \mathbf{n}_3 \end{pmatrix} \end{cases}$$

     - si $d_1 \neq d_2 \neq d_3$ et donc $d' = -d_2$

$$\begin{cases} R' &= \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ \mathbf{t}' &= (d_3 + d_1)\mathbf{n}' \end{cases}$$

     - si $d_1 = d_2 = d_3$ et $d' = -d_2$
       R' est une symétrie d'axe $\mathbf{n}'$

$$\begin{cases} R' &= -I + 2\mathbf{n}'\mathbf{n}'^T \\ \mathbf{t}' &= -2d'\mathbf{n}' \end{cases}$$

5. Eliminer la (les) mauvaise(s) solution(s)

FIG. 1.6 – Un schéma général pour l'estimation du mouvement à partir d'un plan de référence

Chapitre 2

Le suivi d'objets dans une séquence d'images

Sommaire

2.1 Généralités

2.1.1 Qu'est-ce que le suivi ?

Dans tout système biologique de vision, la capacité d'analyse des images est très développée parce qu'elle s'avère être d'une importance capitale dans de nombreuses situations, comme par

exemple la chasse ou la communication. Dans le cas de la vision humaine, le traitement du signal effectué à la fois par les cellules nerveuses de la rétine et par le reste du cerveau permet d'extraire de l'image reconstituée sur la rétine une quantité d'information gigantesque, traitée en temps réel. Lorsque l'on veut suivre un objet, le cerveau effectue différentes tâches : la reconnaissance de l'objet en question, l'estimation de son mouvement .... L'idée de la vision par ordinateur est de construire un système capable de reproduire de telles opérations ; pour cela, on procède par étapes en construisant différents modules pour estimer le mouvement d'objets connus ou inconnus, reconnaître des objets, les suivre dans une séquence d'image, ... L'objectif de ce chapitre est d'explorer un de ces modules, le module du suivi.

L'idée de base d'un module de suivi est de pouvoir suivre un objet (pas forcément connu au départ) le plus longtemps possible. Mais que signifie exactement l'expression « suivre un objet » ? Intuitivement, suivre un objet dans une séquence d'images, c'est être capable de localiser cet objet dans chaque image, la caméra et/ou l'objet pouvant être en mouvement. Mais le suivi ne se limite pas à cette tâche de localisation : on doit aussi caractériser l'objet d'intérêt. La caractérisation de l'objet dépend de l'application visée, et peut être une information de taille, d'orientation, de couleur ou de forme. Dans certaines applications, il peut être utile de connaître les déformations possibles de l'objet. L'ensemble de ces informations constituent l'*état* de l'objet d'intérêt à un instant donné. Selon l'application on va choisir de suivre des points d'intérêt, des primitives géométriques (lignes, courbes, ...), ou plutôt une forme ou un contour (dans le cas du suivi d'un visage par exemple). L'objectif du suivi est donc d'inférer un certain nombre de paramètres relatifs à l'état de l'objet d'intérêt afin de pouvoir le localiser et le caractériser dans chacune des images de la séquence.

Certains facteurs rendent le problème plus complexe : du bruit dans les images, des occultations de l'objet d'intérêt ou des changements d'illumination de la scène sont des problèmes récurrents dans les algorithmes en vision par ordinateur, et rendent la tâche du suivi difficile. D'autre part, on ne connaît pas forcément la nature de l'objet à suivre (dans le cas où l'algorithme doit d'abord reconnaître l'objet pour ensuite le suivre) ; il faut donc exploiter l'information disponible qui n'est pas toujours complète, et attendre d'avoir plus d'images pour avoir davantage d'information.

2.1.2 Détection vs Suivi

Comme on l'a dit précédemment, suivre un objet dans une séquence d'images, c'est être capable de localiser cet objet dans chacune des images grâce à certaines caractéristiques que l'on définit au préalable. On pourrait donc en toute légitimité poser le problème du suivi en terme de détection de l'objet dans chacune des images de la séquence. Ce paragraphe a pour but de comparer les approches détection et suivi d'objet, et de montrer pourquoi la détection est souvent inadaptée pour faire du suivi d'objets dans une séquence vidéo.

La détection d'objets dans une image présente plusieurs difficultés. La première difficulté consiste à catégoriser/caractériser ce que l'on veut reconnaître. Le second problème est de savoir comment le reconnaître, autrement dit de savoir quelles techniques il faut employer pour pouvoir effectivement le reconnaître.

Si on connaît avec précision l'objet que l'on veut suivre et que l'on en a un modèle précis, on pourrait le détecter dans chaque image de la séquence et on obtiendrait le suivi de cet objet

dans la séquence. Il faut pour cela être capable de le détecter en toutes conditions (pour toutes orientations, différentes conditions d'illumination ...). Cette approche pour le suivi est simple et attractive, mais est contraignante : il est nécessaire d'avoir un modèle précis de l'objet (pour pouvoir le reconnaître), et du temps. En effet, sans aucune information sur la position de l'objet dans la scène, seule une recherche exhaustive dans chaque image permet de détecter l'objet. Le temps de calcul est donc un élément limitant dans le choix de l'approche pour effectuer le suivi. D'autre part, la détection aboutit généralement à plusieurs réponses ; il faut ensuite effectuer un traitement pour sélectionner parmi l'ensemble des détections la meilleure.

Dans les approches classiques, on utilise un modèle de mouvement pour prédire la localisation de l'objet à l'image suivante. Cela permet de réduire l'espace de recherche et, dans la plupart des cas, l'utilisation de la cohérence temporelle et spatiale améliore la robustesse et la qualité des résultats. Cependant en pratique les problèmes d'occultations, de changements d'illumination ou de bruit dans les données existent toujours et sont difficiles à résoudre, que ça soit dans le cas du suivi par détection ou en utilisant la cohérence temporelle des images.

2.1.3 Applications du suivi

Le suivi d'objets dans une séquence d'images a beaucoup d'applications possibles : en télédétection, le suivi de l'évolution de la couverture au sol permet d'assurer une bonne gestion des ressourses et de planifier les activités. Dans le domaine de la sécurité ou de la surveillance, la vidéo est un moyen efficace de détecter des mouvements, de compter et/ou de suivre des personnes, d'identifier des personnes suspectes, ... Dans le domaine robotique, les applications sont différentes : la vidéo est utilisée pour l'évitement d'obstacles, ou lors de la découverte d'un environnement inconnu, pour la construction d'une carte. Dans le domaine automobile, le suivi sert également à l'élaboration d'algorithmes d'évitement d'obstacles, mais aussi à assister le conducteur dans sa conduite (voire rendre la conduite totalement automatique). Le suivi est également utilisé dans le domaine médical pour faire de la téléopération. Enfin, la Réalité Virtuelle permet à l'utilisateur d'interagir en temps réel avec sa machine, pour des applications de jeux vidéo bien sûr, mais aussi pour des applications industrielles comme par exemple pour pouvoir examiner un lieu sans pour autant être présent sur place, ou encore pour remplacer à moindre coût de nombreux prototypes physiques (qui peuvent alors être testés immédiatement, modifiés si besoin et retestés indéfiniment).

Selon le type d'application, le degré de précision du suivi est plus ou moins grand, et un type de représentation bien choisi permet souvent d'améliorer la qualité du suivi pour obtenir la précision souhaitée en rapport avec l'application.

2.2 Différentes formes de représentation de l'objet

Une des principales difficultés dans le suivi d'objets est le choix de la représentation de l'objet. Il existe de nombreuses représentations, et il est important de faire le bon choix ; les performances du suivi seront d'autant meilleures que le choix de représentation de l'objet sera bon. Ce choix est généralement guidé par l'application visée, le type de capteur utilisé ou le type de scène étudiée.

2.2.1 Les modèles géométriques

On peut représenter un objet par une forme simple (cube, sphère, ...), un ensemble de formes simples (une pyramide posée sur un parallélépipède pour la représentation d'un bâtiment par exemple), ou un modèle plus complexe (une voiture, un phare, ...). Les modèles utilisés sont nombreux et divers ; on peut les classer en trois grandes catégories : les modèles 3D (parallélépipèdes, pyramides ou formes plus complexes), les modèles 2D (carrés, ellipses) et les modèles déformables (définis par exemple par des contours actifs).

La difficulté est de choisir le bon modèle compte tenu de l'application visée. En effet, plus le modèle est simple, plus la classe d'objets correspondant au modèle est grande, et le risque de confondre l'objet d'intérêt augmente. C'est un problème bien connu et étudié, relié au problème de *la mise en correspondance* : il s'agit de trouver la meilleure correspondance entre les données observées et le modèle que l'on a de l'objet. Un moyen de réduire l'ambiguïté associée à la mise en correspondance est d'accroître la complexité du modèle. Une autre possibilité est d'introduire un contexte dans lequel chaque primitive est détectée sans accroître la complexité de la primitive en question. Cette dernière approche est proposée par Wang [WNC04]. Cependant, un modèle trop complexe aboutit souvent à la perte de la cible au cours du suivi dès que l'apparence du modèle change (à cause de changements d'illumination de la scène ou d'occultation de l'objet par exemple).

Le tracker RAPiD de Harris [Har92] fut l'un des premiers tracker 3D à fonctionner en temps réel parce que son implémentation était très peu coûteuse en calculs. Même si depuis de nombreuses améliorations ont été proposées, les composants de base de cet algorithme ont été repris dans de nombreux systèmes actuels. Nous allons donc le décrire brièvement. L'idée générale est de considérer un ensemble de points 3D de l'objet, que l'on appelle *points de contrôle*. Ces points doivent être suceptibles de se projeter sur des contours à forts contrastes dans l'image. On peut donc choisir d'échantillonner ces points le long des contours de l'objet par exemple. L'algorithme permet de retrouver le mouvement 3D de l'objet entre deux images consécutives en mesurant le déplacement 2D des points de contrôle. Une fois initialisé, l'algorithme parcourt une simple boucle : pour chaque image, la pose calculée (qui peut être la pose estimée à l'image précédente) sert à prédire parmi l'ensemble des points de contrôle, ceux qui seront visibles et leur localisation dans l'image. Les points de contrôle sont mis en correspondance avec les contours de l'image, et une nouvelle pose est estimée à partir de ces correspondances. Pour chaque point de contrôle, le système regarde sa projection dans l'image par rapport à sa projection dans l'image précédente. Chaque point de contrôle fournit une équation, et l'ensemble du système permet de corriger la pose prédite. Pour plus de détails sur l'implémentation, on peut se référer au document de référence [Har92], ou à l'étude réalisée par Lepetit et Fua sur le suivi 3D basé modèle [LF05].

Pour citer quelques utilisation de modèles géométriques, Koller [KDN94] utilise des modèles de voitures sophistiqués (la voiture est représentée par un polyèdre 3D, modélisé avec 12 degrés de liberté) pour effectuer du suivi de véhicules dans le but de faire du contrôle du traffic autoroutier. Kass [KWT87] utilise quant à lui un contour actif pour faire du suivi d'objets déformables. Reveret [RC98] utilise une surface paramétrique 3D comme modèle pour des lèvres. L'intérêt de ce type de modèle est de pouvoir être appliqué simultanément à l'analyse et à la synthèse de lèvres qui bougent, dans le cadre du traitement automatique de la parole.

2.2.2 Les modèles articulés

Les objets non rigides sont des objets dont la forme propre varie au cours du temps. C'est le cas de la plupart des objets naturels, végétaux, animaux et humains. Les représentations utilisées pour les objets rigides (modèles géométriques) ne peuvent généralement pas s'appliquer et il faut mettre en place d'autres types de représentation.

Certains choisissent de modéliser un humain par un squelette plus ou moins complexe : 6 segments (2 bras, 2 jambes, un torse et une tête dans les travaux d'Atika en 1984 [Ati84], mais d'autres travaux utilisent un squelette de 17 segments, ou encore dans [Roh94] un modèle volumique de 14 cylindres elliptiques est utilisé. L'utilisation d'un modèle articulé pour le suivi de la main (ou de tout autre partie du corps), qu'il soit squelettique, comme dans [Dor94, LK95], surfacique, comme dans [KH95, HH96], ou volumique, comme dans [DDHF06], permet de traiter de manière simple les problèmes d'auto-occulations.

Le suivi de certaines parties du corps (comme le suivi des mains, du visage ou des lèvres) pour des applications d'interaction homme-machine, est en plein essor. Rehg [RK94] propose un modèle 3D articulé de la main dans le cadre du suivi de la main pour une application de ce type.

2.2.3 Les modèles d'apparence de l'objet

Dans les cas où il est difficile de qualifier l'objet de façon précise (bruit trop important ou objets trop complexes à modéliser), on peut utiliser un modèle d'apparence. Ces modèles peuvent être basés sur la couleur des objets (utilisation d'histogrammes, comme dans [PHVG02, CRM03]), les invariants des niveaux de gris [Sch96], la texture. Des travaux plus récents utilisent des modèles d'apparence actifs (en anglais *Active Appearance Model*), comme dans [CHTH01]. Lorsqu'aucun modèle de ce type n'est disponible mais qu'une collection d'images de l'objet l'est, on peut utiliser cet ensemble d'images comme modèle d'apparence.

L'avantage de ce type de modèle est qu'il permet de faire du suivi avec assez peu d'informations sur l'objet à suivre, ce qui le rend facilement généralisable. Cependant, le fait d'utiliser relativement peu d'informations peut aboutir à un suivi de « moins bonne qualité » que si on avait choisi un modèle plus complexe et réaliste.

L'utilisation de la couleur

De nombreux travaux de suivi utilisent la couleur comme l'information discriminante pour effectuer le suivi. Dans ce cas, on calcule les invariants en niveaux de gris ou un histogramme de couleur de la région à suivre, puis recherché dans les images suivantes de la séquence.

La couleur est une information riche pour des applications dans des domaines variés : le suivi de visage ou de mains [PF97], les joueurs de foot [JC02, LLLP05], en robotique pour l'asservissement visuel du robot.

Les modèles texturés

La texture concerne la manière dont la couleur est organisée sur la surface d'un objet en une répétition d'éléments et ceci indépendamment de la forme. Dans de nombreux cas, c'est une

information riche, qui permet de discriminer facilement l'objet du reste de la scène.

De nombreux travaux utilisant la texture ont donc été réalisés pour suivre un ou plusieurs objets dans une scène : Schödl et Haro dans [SHE98] comparent un modèle texturé au flux d'image entrant pour faire du suivi de la tête. Le problème de l'utilisation de la texture est le coût calculatoire qu'elle engendre : en général les algorithmes proposés ne fonctionnent pas en temps réel.

Jurie et Dhome [JD00] remédient à ce problème et proposent une approche efficace pour suivre le déplacement d'un motif dans une séquence d'images. Ils procèdent en deux étapes : une première étape hors ligne permet l'apprentissage d'une matrice d'interaction qui établit la relation entre la déformation du motif et son déplacement dans l'image. La deuxième étape, itérative, est exécutée en ligne et consiste à prédire la position du motif dans l'image, puis en utilisant la différence entre le motif observé et le motif prédit et la matrice d'interaction, à calculer un vecteur correctif sur la position prédite. Les calculs effectués lors de cette deuxième étape sont très peu coûteux et permettent à l'algorithme de tourner en temps réel. L'inconvénient de cette méthode est la non gestion des occultations (partielles ou totales), ce qui implique d'être dans un environnement très contrôlé. Jurie et Dhome ont proposé pour remédier à ce problème une approche de suivi 3D d'objets [JD01] (extension de l'approche 2D précédente), puis un algorithme plus général, avec une gestion des occultations partielles [JD02].

Pour gagner en robustesse, certains combinent des primitives de type contour et d'autres de type texture pour effectuer le suivi. Par exemple dans [PM05], un modèle hybride est utilisé pour faire du suivi temps réel : des primitives de type contours sont combinées à des primitives texturées pour assurer un suivi de meilleure qualité que s'il était effectué avec un seul des deux types de primitives.

Les modèles constitués d'images de l'objet

Dans certains cas où aucun des modèles précédemment décrit ne convient pour modéliser l'objet d'intérêt, on peut représenter notre objet d'intérêt 3D par un ensemble d'images 2D, appelées *images de référence*, qui représentent l'apparence de l'objet à suivre. C'est l'approche utilisée par Duculty dans [DDJ02] pour faire du suivi temps réel de visages. L'approche utilise des matrices d'interaction calculées durant une étape hors ligne pour suivre l'image de référence dans l'image courante. Lorsqu'il y a des changements d'apparence, l'algorithme change l'image de référence et choisit celle qui minimise un critère basé sur les différences de niveaux de gris entre les images de référence et l'image courante. L'approche est efficace, mais implique d'avoir des images de plusieurs points de vue de l'objet. Cette représentation est donc spécifique à un seul objet et ne peut être utilisée dans un cadre plus général, pour suivre des objets différents par exemple. Dans la pratique, on n'a pas toujours la possibilité d'avoir des images de l'objet avant d'avoir la séquence, et dans ce cas, cette approche ne peut pas être utilisée.

Le modèle du fond

Le dernier modèle d'apparence que nous présentons ici ne concerne pas l'apparence de l'objet d'intérêt mais l'apparence du reste de la scène. Cette modélisation n'est directement possible que si la caméra utilisée est fixe et à condition que l'objet d'intérêt ne soit pas dans la scène dès d'initialisation. Dans ce cas, la méthode fonctionne de la manière suivante : la première image (ne contenant pas l'objet d'intérêt) sert de modèle ; pour chaque image de la séquence, on

soustrait l'image à l'image de référence (le modèle). Le résultat obtenu correspond à l'apparition des éléments nouveaux dans la scène. Si la scène est fixe et que seul l'objet d'intérêt est en mouvement, il n'est pas forcément utile de mettre à jour le modèle. Par contre, si la scène change au cours du temps, on peut à chaque image de la séquence, faire la soustraction par rapport à l'image de référence puis mettre à jour le modèle.

2.2.4 Les modèles avec marqueurs

Il y a une dernière catégorie de représentation de l'objet d'intérêt : les modèles avec marqueurs. L'utilisation de marqueurs n'est pas toujours possible, mais lorsqu'elle l'est, s'avère efficace. Les marqueurs constituent la plupart du temps des primitives faciles à extraire dans les images, et fournissent des mesures fiables et faciles à exploiter pour l'estimation de la pose par exemple.

Ces marqueurs sont depuis longtemps utilisés par les photogrammètres qui ont besoin de mesures précises pour reconstruire des objets en 3D. La localisation des marqueurs dans les images est bien plus précise que celle obtenue par extraction de points d'intérêt. Aujourd'hui, les marqueurs sont utilisés surtout pour les applications en réalité virtuelle et la capture du mouvement.

Ainsi, des marqueurs circulaires de couleur sont utilisés dans [NC96] pour détecter des points particuliers avec précision. Un autre exemple est celui de l'ARToolKit [KB99] qui utilise des régions de motif carrés noirs et blancs comme marqueurs.

L'utilisation des marqueurs se fait pour des applications de réalité augmentée (chirurgie assistée par ordinateur et imagerie médicale en général), et de navigation de robots. En robotique, la reconnaissance de repères par vision passive simplifie le matériel et abaisse les coûts comparativement aux systèmes de suivi utilisant le laser, le sonar ou autre équipement de vision active. En imagerie médicale, les marqueurs permettent, une fois localisés avec précision, de reconstruire un modèle 3D virtuel de l'organe examiné, et cela procure une assistance visuelle performante pour le chirurgien. L'utilisation de ces marqueurs permet également de controler les corrections chirurgicales mises en place.

2.3 Les informations extraites des images

Une fois que le choix de la représentation de l'objet est effectué, il faut pouvoir extraire des éléments des images pour comparer ces éléments au modèle qui a été choisi. Les informations que l'on peut extraire d'une image sont nombreuses, et le choix est souvent guidé par le modèle choisi précédemment ; cependant plusieurs types d'informations peuvent être utilisés, voire combinés pour rendre la comparaison plus fiable et le suivi plus robuste. Cette partie a donc pour objectif de présenter les différents types d'information que l'on peut extraire des images, mettant en évidence les points forts et les faiblesses de ces informations pour certaines applications.

2.3.1 Les marqueurs

Comme on l'a dit précédemment, les marqueurs sont les informations les plus faciles à extraire des images et les plus fiables. La connaissance de trois marqueurs sur un objet rigide est suffisante pour calculer sa position et son orientation. Bien sûr, l'occultation des marqueurs est un phénomène inévitable. Une solution pour pallier au problème est d'augmenter le nombre de marqueurs (pour espérer en avoir toujours au moins trois visibles), ou d'augmenter le nombre de

caméras pour avoir davantage de points de vue et de visibilité des marqueurs.

Les domaines d'applications qui utilisent des marqueurs sont divers : en robotique, ils permettet la navigation du robot. En imagerie médicale, ils permettent de recaler des images de types différents pour fusionner les données et faciliter les opérations de chirurgie par exemple. D'autres domaines, comme celui de la capture du mouvement humain ou la réalité augmentée, utilisent aussi beaucoup les marqueurs. Dans notre cadre de travail (comme dans beaucoup d'autres cas), il est impossible d'utiliser des marqueurs. Nous ne nous étendrons donc pas plus sur ce type d'information.

2.3.2 Les contours

Les contours constituent des indices riches (au même titre que les points d'intérêt, voir paragraphe suivant) pour toute interprétation d'une image. La détection de contours est donc souvent une des premières étapes de la vision par ordinateur. Pour cela, aucune information *a priori* sur les objets présents dans l'image n'est utilisée.

Un contour est une discontinuité présente dans l'image. En pratique, les contours représentent des variations locales et rapides de la fonction d'intensité dans l'image. Ces variations peuvent provenir de différentes sources qui sont :

- *Géométriques* : dans ce cas, on a les discontinuités physiques sur la surface de l'objet et les frontières physiques qui séparent l'objet du fond de la scène. Les arêtes détectées sont donc les « vrais » contours de l'objet.
- *Photométriques* : dans ce cas, on a les discontinuités de l'illumination et des propriétés de réflectance de la surface de l'objet. Les arêtes détectées correspondent à des ombres, à des textures ou à des reflets provoqués par des régions spéculaires de la scène.

L'objectif de la détection de contours dans une image n'est pas de détecter toutes ces transitions, mais plutôt celles qui permettent une description pertinente de l'image. Une bonne détection de contours facilite la mise en correspondance d'entités (en vue de l'estimation du mouvement par exemple). Un des problèmes fondamentaux est la résistance aux bruits et aux petites variations locales dues à la texture ou à des irrégularités de surface. Pour cela, on effectue en général un prétraitement : passer un filtre sur l'image permet de lisser ces petites irrégularités locales. De nombreux filtres existent, on peut les classer en deux catégories. D'une part les filtres linéaires (un filtre est *linéaire* si la nouvelle valeur du pixel de l'image résultat est une combinaison linéaire des valeurs des pixels situés dans une fenêtre de l'image initiale), comme par exemple le filtre moyenneur. D'autre part les filtres non linéaires, comme le filtre médian ou d'autres filtres morphologiques ou adaptatifs.

Pour la détection de contours proprement dite, les premières idées de modélisation étaient très simples. Une discontinuité était assimilée à un fort gradient ou à un passage par zéro du Laplacien [1]. Nous allons présenter brièvement ces deux approches, puis présenter ensuite le modèle de Canny.

[1] En calcul vectoriel, le laplacien est un opérateur différentiel égal à la somme de toutes les deuxièmes dérivées partielles non mixtes d'une variable dépendante

Pour calculer le gradient discret d'une image, les techniques sont nombreuses. Dans la plupart des cas, on utilise un produit de convolution. Les masques les plus utilisés sont ceux de Roberts, Prewitt et Sobel [CP95]. Lorsque le gradient est calculé, un simple seuillage fournit de bonnes informations sur la présence des discontinuités. Cependant, les contours obtenus sont souvent trop épais, et il faut les affiner. De nombreuses techniques existent : l'application de filtres morphologiques (érosion, dilatation, calcul de squelette) est intéressante mais imprécise pour la localisation. L'approche la plus répandue est le seuillage par hystérésis proposée à l'origine par Canny en même temps que son détecteur [Can86]. Cette technique consiste à choisir deux seuils, $s1$ et $s2$ et à effectuer un chaînage en opérant une sélection des gradients de la manière suivante : l'image est parcourue de haut en bas et de gauche à droite. Dès qu'une valeur de gradient dépasse le seuil $s1$, elle constitue le départ de la chaîne. On essaie alors de continuer la chaîne dans les deux sens, en fontion de l'orientation du gradient. Tant que le gradient sur le pixel est supérieur au seuil $s2$, on continue la chaîne, sinon une extrémité de la chaîne est atteinte. Lorsqu'une chaîne est terminée, le parcours de l'image reprend là où il s'était arrêté. Nous utilisons ce filtre pour la segmentation de nos images ; la méthode ainsi que l'implémentation sera reprise en détails au chapitre 4.

L'autre méthode pour détecter les variations locales consiste à détecter les passages par zéro du Laplacien ou de la dérivée seconde dans une direction donnée. En effet, les extrema du gradient correspondent aux passages par zéro de la dérivée de celui-ci, le Laplacien. Cependant, pour ne pas être sensible aux valeurs minimales du gradient, c'est-à-dire les valeurs nulles qui correspondent alors à des zones homogènes, il faut détecter les passages par zéro du Laplacien et non pas seulement les valeurs nulles de celui-ci. Le problème de l'utilisation du Laplacien est sa sensibilité aux petites variations, dues la plupart du temps au bruit. Pour résoudre ce problème, différentes approches ont été proposées ; certains opèrent un fort lissage de l'image, par exemple avec un filtre gaussien. D'autres, comme Marr, proposent de convoluer l'image avec un masque approchant la dérivée d'une gaussienne.

Dans la littérature, on trouve beaucoup d'autres détecteurs de contours. Nous n'allons pas ici énumérer toutes les techniques, mais simplement citer quelques travaux : Bergholm [Ber87] propose une détection de contours pyramidale, Kundu [Kun90] une classification statistique des pixels pour obtenir les contours. Les champs de Markov a été largement utilisée aussi : le principe est d'associer à chaque pixel une fonction d'énergie qu'il faut minimiser. D'autres détecteurs utilisent les réseaux de neurones, les lignes de crête [Che92], ou la transformée de Hough [DH72] pour détecter des formes particulières dans les images.

2.3.3 Les points d'intérêt

Les points d'intérêt correspondent à des doubles discontinuités de la fonction d'intensité. Celles-ci peuvent être provoquées, comme pour les contours, par les discontinuités de la fonction de réflectance ou des discontinuités de profondeur. Ce sont par exemple les coins, les jonctions en T ou les points de fortes variations de texture.

Les avantages d'utiliser des points d'intérêt sont nombreux : présents dans une grande majorité d'images, ils sont robustes aux occultations (ils sont soit visibles soit occultés complètement). D'autre part, il n'y a pas d'opération de chaînage à réaliser (comme il faut le faire pour les contours), et ils constituent une information plus fiable que les contours parce que les contraintes sur la fonction d'intensité sont plus nombreuses. Ils sont donc très souvent utilisés.

La détection des points d'intérêts se fait en cherchant les doubles discontinuités de la fonction d'intensité. De nombreuses méthodes ont été proposées. On distingue deux grands types d'approches : les approches « contours » détectent les contours dans l'image, puis extraient les points d'intérêt le long des contours (intersection de contours ou courbure maximale). Les approches « intensité » regardent directement la fonction d'intensité dans les images pour extraire directement les points d'intérêt. Ce sont les approches de ce type qui sont les plus utilisées, en raison de leur indépendance vis à vis de la détection de contours, et vis à vis du type de point d'intérêt. Plusieurs détecteurs existent, notamment celui de Förstner [FG87], de Harris [HS88], de Susan [SB95] ou de Shi-Tomasi [ST94].

L'opérateur de Förstner

Ce détecteur permet d'extraire des points d'intérêt de différents types (coins ou jonctions) sans passer par une extraction de contours. C'est un opérateur local, qui fonctionne de la manière suivante : il s'agit d'abord de trouver des fenêtres candidates dont on fait l'hypothèse qu'elles contiennent chacune un seul point d'intérêt. Ce point d'intérêt est ensuite localisé dans la fenêtre et caractérisé. Une mesure de confiance est obtenue à cette étape, et permet de rejeter ou d'accepter l'hypothèse qui avait été faite. Il y a donc 3 étapes :

1. *Détection* : sélection des fenêtres candidates. La sélection est basée sur la valeur moyenne de l'intensité du gradient dans une fenêtre d'intensité donnée. On détecte les maxima locaux de cette fonction.
2. *Caractérisation* : classification de la fonction image dans les fenêtres. La classification fait la distinction entre les différents types de points singuliers : coins, anneaux, spirales et textures isotropes (se fait par un test statistique).
3. *Localisation* : estimation du point optimal dans les fenêtres.

Le détecteur de Harris

C'est une méthode différentielle, qui se fonde sur l'analyse de la variation de la luminance au voisinage d'un point. La matrice d'autocorrélation, notée Ξ représente la variation locale de l'image en un point. Un point sera considéré comme un coin si pour tous les déplacements $(\Delta x, \Delta y)$, la quantité $(\Delta x, \Delta y)\Xi(x, y)(\Delta x, \Delta y)^T$ est grande.
L'algorithme procède en différentes étapes :

1. Calculer les images de gradient selon les deux axes de l'image
2. Pour chacun des points de l'image
   - Estimer la matrice de covariance du gradient et extraire les valeurs propres
   - Si la valeur propre minimale est supérieure à un seuil, c'est un coin.
3. Trier les coins.
4. En parcourant les coins, si il y en a plusieurs dans un même voisinage, ne garder que celui qui a le meilleur score.

Nous invitons le lecteur à se référer à [HS88] pour plus de détails sur l'implémentation de ce détecteur.

Le détecteur de Susan

Ce détecteur considère qu'au niveau d'un coin, on est en présence de deux objets de luminance différente qui déterminent deux zones distinctes dans le voisinage du coin. On détermine la valeur du point d'intérêt par la proportion de points du voisinage qui sont de même luminance : pour un coin à angle droit, on obtient un quart des points, et pour un coin plus aigu ou plus obtu, on obtient au plus la moitié des points, cas qui correspondrait à un contour.

L'intérêt de ce détecteur est qu'il est très simple à mettre en œuvre, ce qui lui donne une rapidité bien supérieure aux autres. Cependant il est moins robuste à la compression.

Les détecteurs classiques ont été longtemps utilisés pour détecter les points d'intérêt. Depuis quelques années, beaucoup de travaux ont été réalisés dans le but d'utiliser au mieux ces détections pour caractériser les points détectés ; les descripteurs locaux permettent aujourd'hui de caractériser plus précisément les points d'intérêts, et sont très utilsés pour la reconnaissance, ou l'indexation d'image. On peut se référer aux travaux de C. Schmid [Sch96] pour plus de détails.

2.3.4 Les régions

On peut choisir d'extraire dans l'image des régions. Ces régions, définies par un ensemble de pixels connexes ayant des propriétés communes qui les différencient des pixels des régions voisines. Pour estimer l'homogénéité des régions, plusieurs critères peuvent être utilisés, le plus courant étant la variance des niveaux de gris.

Une caractéristique générale du suivi de régions consiste en des hypothèses d'homogénéité du mouvement à l'intérieur de la région, et d'invariance de l'apparence de celle-ci dans le temps.

Par exemple, Bascle et Deriche [BD95] proposent une approche pour suivre des formes complexes sur une séquence d'images. L'utilisation de l'information donnée par les niveaux de gris (par rapport à l'information de contour) améliore les performances du suivi de modèles déformables en présence de texture.

2.3.5 Le mouvement

Le mouvement est une information importante lors de l'analyse d'une séquence d'images. Il existe différentes méthodes pour quantifier le mouvement qui existe entre deux ou plusieurs images d'une séquence. Une des plus connues est le flot optique [HS81] qui consiste à calculer le champ de vitesses (norme et direction) mesurant le déplacement apparent des points dans la scène.

Le flot optique

Le calcul du flot optique consiste à extraire un champ de vitesses dense à partir d'une séquence d'images, typiquement en faisant l'hypothèse que l'intensité (ou la couleur) est conservée au cours du déplacement. De nombreuses techniques ont été proposées pour le calcul du flot optique, que Barron, Fleet et Beauchemin [BFBB] classent en quatre catégories :

1. Les méthodes différentielles, qui calculent la vitesse à partir des dérivées de l'intensité dans l'image ou en utilisant des images préfiltrées.

2. Les méthodes basées sur la corrélation, qui consistent à chercher une correspondance entre des fenêtres d'images de deux instants consécutifs.
3. Les méthodes basées sur l'énergie, qui se basent sur la recherche de la solution pour un instant donné t comme un minimum d'une fonctionnelle.
4. Les méthodes basées sur la phase, qui consistent à faire une analyse fréquentielle locale sur des portions d'espace-temps de la séquence d'images.

Le mouvement par différence d'images

On se place dans le cas où la caméra est statique et la scène contient un ou plusieurs objets en mouvement. Ces méthodes ne donnent pas une mesure du mouvement au sens de vecteurs de vitesse et direction de déplacement des objets, mais permettent de recueillir des indices sur les endroits de l'image où il y a eu un déplacement.

Le principe général est de calculer la différence entre deux (ou plusieurs) images. Lorsqu'il n'y a pas recouvrement entre les deux positions, l'information du signe de la différence permet de séparer les objets. Par contre, lorsqu'il y a recouvrement, seules les zones en périphérie de l'objet permettent de détecter le mouvement. Dans ce cas, la différence apporte un indice visuel qu'il faut coupler avec d'autres informations pour reconstruire la scène complète.

Cette méthode de différence d'images est utilisée depuis de nombreuses années dans les travaux sur la segmentation de mouvement, comme dans [Jai81] ou [Tho80]. Plus récemment, certains travaux combinent cette information de différence avec des informations statiques. Leung et Yang recherchent des contours communs entre une image de différence et une image de gradient pour identifier les mouvements d'une personne [LY95]. D'autres travaux existent ; dans tous les cas, cette méthode de différence d'images couplée avec d'autres sources d'information s'est avérée utile pour détecter des objets déformables.

Nous ne nous étendrons pas plus sur ce type d'approche. Nos travaux se placent dans un contexte où la caméra est en mouvement, et nous n'utilisons pas ces méthodes pour l'estimation du mouvement. Dans notre cas, il s'agit de déterminer le mouvement 3D de la caméra et non pas le mouvement apparent des objets dans les images.

2.3.6 Les histogrammes de couleur ou de niveaux de gris

On appelle *histogramme* de l'image I la fonction H définie sur l'ensemble des entiers naturels par

$$H_I(i) = Card\{P : I(P) = i\}$$

.

On trouve également la définition suivante : pour une image I de taille $N \times M$

$$H_I(i) = \sum_{x=1}^{N} \sum_{y=1}^{M} \delta(I(x,y) - i)$$

Lorsque l'on calcule des histogrammes sur des images ou des régions d'image, il faut pouvoir ensuite les comparer. Pour cela, il existe différentes mesures de similarité, parmi les plus connues :

- La distance Euclidienne

$$D(A,B) = \sqrt{\sum_i (H_A(i) - H_B(i))^2}$$

- Les mesures définies à partir des fonctions de densité de probabilités. Ce sont les plus utilisées dans la littérature. Parmi elles, on trouve
  - La distance K-L (Kullback-Leibler) [KL51]

$$D(A,B) = \sum_i P_B(i) log \frac{P_B(i)}{P_A(i)}$$

où $P_A(i)$ représente la probabilité d'avoir l'élement i dans la distribution A. Si les histogrammes sont normalisés, on peut écrire :

$$D(A,B) = \sum_i H_B(i) log \frac{H_B(i)}{H_A(i)}$$

  - La K-divergence [Lin91]

$$D(A,B) = \sum_i P_B(i) log \left(P_B(i) \left(\frac{1}{2P_A(i)} + \frac{1}{2P_B(i)} \right) \right)$$

De même on a, si les histogrammes sont normalisés

$$D(A,B) = \sum_i H_B(i) log \left(H_B(i) \left(\frac{1}{2H_A(i)} + \frac{1}{2H_B(i)} \right) \right)$$

  - La distance de Bhattacharyya [Bha43]

$$D(A,B) = -log \sum_i \sqrt{P_B(i) P_A(i)}$$

et pour des histogrammes normalisés

$$D(A,B) = -log \sum_i \sqrt{H_B(i) H_A(i)}$$

  - La distance de Matusita [Mat55]

$$D(A,B) = \sqrt{\sum_i \sqrt{P_B(i)} - \sqrt{P_A(i)}}$$

et pour des histogrammes normalisés

$$D(A,B) = \sqrt{\sum_i \sqrt{H_B(i)} - \sqrt{H_A(i)}}$$

- Test du χ^2 : pour mesurer la similarité entre les histogrammes A et B, on calcule :

$$\chi^2(A,B) = \sum_i \frac{(H_A(i) - H_B(i))^2}{H_A(i)}$$

Sur des images couleur, il est possible de choisir des espaces de couleur dans lesquels certaines composantes sont invariantes. L'espace RVB est non invariant aux variations de luminance, mais si on normalise les couleurs par la luminance, on obtient l'invariance à la luminance ($r = \frac{R}{R+G+B}$ et pareil pour les 2 autres composantes).

Le calcul d'un histogramme sur une région de l'image étant une opération simple, les histogrammes sont souvent utilisés pour des applications qui doivent fonctionner en temps réel. On peut se reporter aux travaux de Meer [CRM03], Zivkovic [ZK04] ou à nos travaux [JSR05] pour des exemples d'applications. Un autre avantage de l'utilisation d'histogrammes est son invariance aux rotations.

Dans la plupart des applications, l'histogramme est calculé sur une région d'intérêt, initialisée avec une forme de géométrie simple (rectangle, ellipse). La région d'intérêt englobe l'objet d'intérêt (au centre) et du fond. Pour limiter l'influence du fond et privilégier l'information du centre de la zone, on calcule un *histogramme pondéré*. Plus les pixels sont loin du centre, moins ils ont de poids dans l'histogramme final. Pour une image I de dimensions $N \times M$, on définit un histogramme pondéré de la manière suivante

$$H_I(i) = \sum_{x=1}^{N} \sum_{y=1}^{M} w(x,y)\delta(I(x,y) - i)$$

où w est une mesure locale au voisinage de chaque pixel

Cependant, l'utilisation des histogrammes présente deux inconvénients majeurs. Le premier est qu'un histogramme perd toute l'information spatiale des pixels dans l'image. Il permet de connaître la répartition des couleurs de l'image mais ne prend pas en compte la couleur d'un pixel par rapport à ses voisins. Et le deuxième est que la détermination du nombre de classes d'un histogramme est délicate. Un trop faible nombre de classes fait perdre de l'information et aboutit à supprimer les différences pouvant exister entre des groupes de l'ensemble étudié. A l'inverse, un trop grand nombre de classes aboutit à des graphiques incohérents où certaines classes deviennent vides ou presque car la taille de l'échantillon est fini. Cela pose des problèmes importants : en cas de bruit dans les données, l'histogramme peut se trouver « translaté » d'une classe par rapport au modèle ; les deux histogrammes ne seront plus du tout similaire.

2.4 La stratégie du suivi

Après avoir choisi comment modéliser notre objet et défini les caractéristiques à extraire des images, il reste à trouver la stratégie qui permettra de suivre notre objet avec les meilleures performances. Comment peut-on gérér les occultations de l'objet d'intérêt ou initialiser le suivi ? Faut-il mettre à jour le modèle ? Si oui, à quel moment ? Dans cette section, nous allons essayer de donner des éléments de réponse à ces questions, en mettant en évidence les forces et faiblesses des méthodes existantes.

2.4.1 Les occultations

L'utilisation d'une unique caméra est à l'origine de nombreux problèmes, notamment liés au fait que l'on utilise des informations en entrée en deux dimensions (les images), alors que la scène est en réalité 3D (voire 3D + t si la scène change au cours du temps). Une certaine

quantité d'informations est alors perdue lors de la projection de la scène dans le plan image. Les occultations sont un des problèmes majeurs liés à cette projection.

La gestion des occultations dans un algorithme de suivi d'objets est un des problèmes les plus difficiles. Si un modèle de détection est intégré à l'algorithme de suivi, il pourra être utilisé pour retrouver l'objet une fois qu'il réapparaîtra dans l'image. Dans le cas contraire, l'estimation du mouvement dans la scène permet de résoudre en partie le problème, en cherchant l'objet occulté dans une direction particulière dans l'image. Cependant, le modèle de mouvement ne permet pas de gérer efficacement les occultations de « longue durée » : plus les images se succèdent sans pour autant retrouver l'objet d'intérêt et plus l'incertitude sur la zone de recherche à couvrir augmente ; dans ce contexte, la probabililté de retrouver l'objet à la fin de l'occultation devient rapidement très faible.

A notre connaissance, il n'existe pas de solution efficace à ce problème d'occultation dans le contexte d'un suivi monoculaire. Pourtant la capacité de traiter les occultations est essentielle à l'utilité d'un système de suivi d'objets dans la plupart des applications. Cette mauvaise gestion (voire non gestion) des occultations provoque l'échec du suivi ; une opération de réinitialisation du système est très souvent nécéssaire.

Pour résoudre le problème, on peut quand même se poser les questions suivantes :
- Concernant l'origine de l'occulation : l'objet est-il occulté par un objet fixe de la scène ou par un objet mobile ?
- Quantitativement, de quelle quantité (en terme de surface) l'objet est-il occulté ? Est-il partiellement ou totalement occulté ?

Les choix algorithmiques seront alors orientés par les réponses à ces deux questions. Dans la littérature, les solutions proposées pour la gestion des occultations sont nombreuses. Certains utilisent l'information donnée par plusieurs caméras pour résoudre le problème [Doc01, DT01]. D'autres, dans le cas du suivi à partir d'une unique caméra, utilisent des modèles d'apparence qui leur permettent de gérer les occultations partielles [SHT+01, Sen02, NS04, YLS04]. Une dernière catégorie de travaux fournissent une solution implicite au problème d'occultation : ce sont les travaux qui font du suivi simultané de plusieurs objets dans la scène. En général, une piste est associée à un objet et lorsqu'un objet est occulté c'est le plus souvent par un autre objet de la scène (qui est suivi lui aussi). Dans ce cas, les deux pistes se rejoignent et forment une piste unique. Puis, plus tard, les deux objets se séparent et une nouvelle piste s'initialise. Par contre, il n'y a aucun moyen de savoir que la nouvelle piste correspond à un objet qui existait précédemment et qui avait simplement été occulté pendant une certaine durée. Nous reviendrons sur ces considérations dans le paragraphe suivant.

2.4.2 Le problème du suivi de plusieurs objets

Une autre difficulté dans le suivi d'objets dans une séquence d'images est le suivi simultané de plusieurs objets, en particulier lorsque deux de ces objets se croisent. En effet, comment identifier correctement les objets pendant et après l'occultation ?

Ce problème est un problème mal posé : comme on l'a évoqué dans le paragraphe précédent, si les objets ont les mêmes caractéristiques, l'algorithme n'a aucun moyen de ne pas confondre les deux objets s'ils se croisent. Et c'est une chance en quelque sorte si l'algorithme continue à suivre

chaque objet avec la bonne identification. Pour résoudre ce problème, on utilise le mouvement (surtout la direction) des objets pour garder la « bonne étiquette » sur les objets qui se croisent. Dans le cas où il n'y a pas de changement brutal de direction et où les objets proviennent de directions différentes, le suivi se passe généralement bien. Les cas les plus difficiles sont ceux pour lesquels les objets proviennent de la même direction et se croisent, ou les cas de changements brutaux de direction des objets : un des exemples classiques est celui de l'applaudissement. Pour résoudre le problème dans le cas où les mouvements sont bien connus, on peut apprendre des mouvements typiques, et suivre efficacement les objets.

2.4.3 La mise à jour du modèle

La mise à jour du modèle est un problème difficile. Dans le cas du suivi d'un objet, lorsque le modèle 3D de l'objet est parfaitement connu, il n'y a pas de problème de mise à jour : une simple comparaison avec le modèle suffit pour attribuer une note de confiance au suivi réalisé. Le problème de la mise à jour du modèle se pose lorsque l'objet suivi n'est pas connu par avance, ou que l'on en a un modèle général et le plus souvent trop simple (un histogramme de couleur, quelques images ...). En effet, dans ce cas, dès que la séquence est longue et que les points de vue de l'objet sont éloignés, il faut pouvoir mettre à jour le modèle pour être capable de gérer les changements d'apparence.

Lorsque la mise à jour du modèle est indispensable, les techniques adoptées sont nombreuses : dans certains cas, le modèle est mis à jour à chaque image, considérant que le suivi d'une image à l'autre se passe bien et que l'on peut faire confiance aux résultats. C'est l'approche utilisée par exemple par [HS90]. Le problème de cette technique est que si pour une image le suivi est mauvais, le modèle change pour prendre en compte ces mauvais résultats, ce qui provoque un très mauvais suivi par la suite .... D'autres, comme par exemple Babu [BPB05], mettent à jour le modèle de manière adaptative ; ils intègrent de l'information des images précédentes. Une autre solution au problème de mise à jour du modèle est de le mettre à jour uniquement lorsque la note de vraisemblance entre les observations et le modèle est suffisamment bonne (c'est-à-dire au-dessus d'un seuil fixé au préalable par l'utilisateur ou en dur dans l'algorithme). C'est ce que Numiaro et al. [NKMG02] font dans leur approche. Enfin, une autre solution est de dire que le modèle de l'objet est bon tant que la note de vraisemblance est suffisamment bonne et donc que la mise à jour du modèle n'est pas nécessaire ; par contre lorsque la note n'est plus assez bonne (il y a donc aussi ici un seuil à fixer), on met à jour le modèle. Même si nous ne sommes pas convaincus que ce soit la meilleure approche pour la mise à jour du modèle, c'est pourtant celle que nous avons choisie [JSR05].

Il n'a pas été prouvé qu'une méthode soit bien plus efficace que les autres : dans un cas, l'algorithme diverge progressivement (pour une mise à jour à chaque image), dans l'autre des problèmes d'occultations peuvent provoquer une mauvaise mise à jour du modèle. D'une manière générale, les performances dépendent surtout de la séquence et de la sélection d'un modèle adapté pour la représentation de l'objet à suivre.

2.4.4 L'initialisation

L'initialisation est déterminante, tant pour la précision que pour la robustesse du suivi. On trouve de nombreuses techniques pour initialiser un algorithme et parmi les plus courantes :

L'initialisation manuelle

On sélectionne dans la première image la région d'intérêt (ou plus généralement une région contenant l'objet d'intérêt). Lorsque le modèle intègre de l'information provenant du fond de la scène, ou si la forme présupposée n'est qu'approximativement adaptée à l'objet, le modèle obtenu n'est pas seulement représentatif de l'objet suivi mais est également influencé par l'environnement immédiat de la cible. Il est donc important lors de la sélection manuelle de l'objet dans la première image de définir une région sans trop de fond, et englobant l'objet d'intérêt. Dans le cas de l'approche de Meer [CRM03], la région d'intérêt est modélisée par une ellipse dans laquelle est calculée un histogramme de couleur. Pour limiter l'influence du fond et privilégier l'information contenue au centre de l'ellipse, il effectue une pondération spatiale des pixels selon leur distance au centre de l'ellipse. Mais cette atténuation n'est pas toujours suffisante.

L'initialisation semi-automatique

On règle à la main certains paramètres et d'autres se règlent automatiquement. Par exemple, dans notre approche [JSR05], la sélection de la zone d'intérêt se fait à la main, mais les seuils et la sélection du nombre de classes de l'histogramme est automatique. On explicitera nos choix avec précision au chapitre 4.

L'initialisation automatique

L'initialisation automatique d'un algorithme de suivi est un problème bien distinct du suivi en lui-même. En général, on fait une détection des objets d'intérêt potentiellement présents dans l'image ; cela implique d'avoir un modèle assez précis de l'objet que l'on veut suivre. Cette phase de détection peut être suivie d'une phase de vérification pour élimier les fausses alarmes éventuellement présentes. On peut aussi procéder de la manière suivante : on commence par détecter un blob de couleur et, dans un deuxième temps, effectuer un ajustement d'un modèle géométrique plus précis.

Une autre solution serait d'avoir une caméra statique et l'objet d'intérêt en mouvement. Par soustraction de fond, on pourrait détecter l'objet d'intérêt. Et, une fois détecté, on pourrait lancer l'algorithme de suivi proprement dit pour des configurations générales (caméra fixe ou en mouvement, objet d'intérêt fixe ou en mouvement). L'intérêt majeur de ce type d'approche est la possibilité de faire du suivi sans aucun modèle explicite de l'objet à suivre, ce qui rend cette approche applicable à tout type d'objet. Il n'y a aucune condition à remplir, que ce soit au niveau de la taille ou de la forme de l'objet, ou sur le type de fond.

2.4.5 L'estimation de l'état de l'objet

Après avoir choisi un modèle de représentation, défini les caractéristiques à extraire des images, et étudié la gestion des difficultés liées au suivi (initialisation, mise à jour du modèle, bruit, occultations), il reste à déterminer comment le système va estimer au cours du temps l'état du système à partir des mesures et des observations disponibles. Les méthodes sont nombreuses, mais certaines sont limitées à certaines hypothèses faites sur le bruit ou sur d'autres paramètres du système, et ne peuvent pas forcément s'adapter à un cadre général.

La mise en correspondance

La mise en correspondance consiste à localiser dans les différentes images, les projections de la même entité 3D de la scène. La qualité de l'appariement obtenu est déterminant pour la suite des traitements, que ce soit de la reconstruction 3D de la scène, de l'estimation de mouvement ou du suivi d'objet. Le processus de mise en correspondance est un problème délicat parce que de nombreux facteurs peuvent entrainer des résultats erronés. C'est le cas notamment du bruit, des changements de luminosité, des occultations et aussi des zones uniformes dans les images qui correspondent à des zones dans lesquelles les pixels peuvent être facilement appariés avec des pixels proches mais qui ne correspondent pas. Pour faire face à ces difficultés, de nombreuses méthodes ont été mises en place ; ce paragraphe a pour but de présenter les différentes méthodes existantes en mettant en évidence leurs points forts et leurs faiblesses, sans toutefois prétendre à faire un état de l'art exhaustif de l'ensemble de ces méthodes.

D'une manière générale, la mise en correspondance repose sur la définition d'un critère et d'une méthode de recherche :

1. le critère peut être un critère d'unicité, de corrélation ou d'invariant. Il permet de définir ce que l'on va comparer dans les images (niveaux de gris, région ...)
2. la méthode de recherche permet de définir l'algorithme de recherche du correspondant dans l'autre image. Les méthodes sont nombreuses, on peut citer parmi les plus courantes
   - La mise en correspondance hiérarchique, qui utilise une pyramide d'images à différentes résolutions. La recherche se fait depuis l'image basse résolution à l'image de plus haute résolution.
   - La mise en correspondance par programmation dynamique
   - La mise en correspondance par relaxation : on cherche d'abord un ensemble de candidats potentiels ; dans un deuxième temps, on cherche parmi cet ensemble le meilleur candidat possible

Le schéma général d'un algorithme de suivi de points d'intérêt ou de régions peut se décomposer en trois grandes étapes :

1. Pour chaque point, on trouve le point le plus proche dans l'image suivante ; la recherche s'effectue selon le critère choisi
2. On compare au « template » de départ ; en général, la corrélation croisée est la méthode utilisée pour effectuer la comparaison des deux régions.
3. On met à jour le « template »

Bien sûr, il est possible d'utiliser d'autres méthodes pour le suivi de points d'intérêt ou de régions : le filtre de Kalman par exemple est très souvent utilisé (voir le paragraphe suivant).

Suivi par corrélation

Bien que très proche de la méthode présentée ci-dessus, nous allons présenter le suivi par corrélation dans un paragraphe spécifique.

La corrélation croisée est une mesure permettant d'évaluer la ressemblance entre deux pixels de deux images. Les travaux de Moravec [Mor80] sont un exemple du suivi par corrélation ; à partir de la dernière position connue de la cible, on recherche sa nouvelle position. Pour cela, on

compare l'imagette (la zone de l'image définie comme cible sur la dernière image) à des zones de taille similaire dans la nouvelle image. La zone qui lui ressemble le plus est choisie comme nouvelle position de l'objet suivi. Il existe plusieurs formules pour calculer la ressemblance entre deux régions ; dans la plupart des cas, cette mesure est basée sur les comparaisons des valeurs des pixels sur les deux régions à comparer.

Le suivi par corrélation est une procédure simple à mettre en œuvre et rapide en terme de calculs, ce qui rend cette approche attrayante. Cependant, le suivi de ce type présente quelques inconvénients : tout d'abord la mesure de ressemblance basée sur une comparaison des valeurs des pixels dans les images est peu robuste aux variations d'illumination dans la scène. Pour la rendre plus robuste, il faut utiliser la corrélation croisée normalisée. C'est d'ailleurs ce qui est fait dans de nombreux travaux. D'autre part, pour des mouvements de rotation, les pixels comparés entre les deux imagettes ne sont plus des pixels qui se correspondent, et bien souvent cela aboutit à la perte de la cible. Une solution à ce problème est de prendre en compte des mouvements de rotation (non seulement des rotations dans le plan image, mais aussi de rotation 3D pour être robuste aux effets de perspective), un autre est de passer à un modèle plus abstrait, par exemple en se basant sur les contours et non plus sur les pixels. Le suivi par contours actifs peut être la solution au problème dans ce cas là.

Suivi par contours actifs

Le premier à présenter ce type d'approche est Kass en 1987 [KWT87] : le principe repose sur la création d'un contour (en anglais : *snake*) formé d'un ensemble de points mobiles et répartis sur une courbe en deux dimensions. Cette courbe qui peut être ouverte ou fermée selon l'application, est placée dans la zone d'intérpet de l'image ou autour d'un objet. Plusieurs équations décrivent son évolution : la courbe se déplace et épouse lentement les contours des objets en fonction de plusieurs paramètres, comme l'élasticité, la tolérance au bruit, ...

Cette dynamique est basée sur la notion d'énergie interne et externe, le but étant de minimiser l'énergie totale présente le long de la courbe. Des contraintes permettent de conserver une courbe lisse avec des points équidistants tout en laissant un certain champ libre pour les déformations. L'énergie interne correspond à la morphologie et aux caractéristiques de la courbe (courbature, longueur, ...). L'énergie externe provient de l'image, les critères sont variables (présence de bords marqués, bruit, ...). L'évolution se fait de manière itérative.

Le schéma général du suivi par contours actifs peut se décomposer en deux étapes :

1. pour chaque nouvelle image, le « snake » précédent sert d'initialisation
2. on laisse évoluer le snake, et on met à jour le modèle

Ce type de méthode présente l'avantage d'être plus flexible quant à la modélisation de l'objet d'intérêt. Elle est beaucoup plus robuste aux variations lumineuses et aux rotations de la cible. Cependant la méthode présente quelques inconvénients : le contour initial doit être initialisé suffisamment proche des objets, le contour actif converge vers un minimum d'énergie local, et l'énergie dépend de la paramétrisation de la courbe et non de la géométrie de l'objet. De nombreuses améliorations ont été proposées, les contours actifs géodésiques en sont un exemple.

Plusieurs méthodes reprenant ce principe ont été formulées en fonction des problèmes à résoudre. Ainsi, les contours actifs sont utilisés en imagerie médicale, ou dans un contexte plus

général pour faire de la reconnaissance de formes ou de la segmentation d'images. Ils sont également utilisés pour faire du suivi d'objets. Lefevre, par exemple, décrit dans [LFMV01] un modèle de contour actif pour le suivi rapide d'objets en mouvement. Plus récemment, il élargit son approche à plusieurs topologies et au suivi multi objets en temps réel [LV05]. Un autre type d'application est celui du suivi de véhicules : [PD99] propose une méthode pour suivre un ou plusieurs objets en utilisant les contours actifs géodésiques.

Notons que cette méthode peut être étendue à d'autres dimensions, en particulier la 3D où la courbe prend la forme d'une enveloppe qui épouse progressivement la surface d'un objet. De telles méthodes sont utilisées pour résoudre des problèmes de segmentation volumique.

Le filtre de Kalman

Le filtre de Kalman est un outil d'estimation classique qui permet de déterminer récursivement l'état d'un système dynamique à partir d'une mesure. Cette méthode est utilisée pour la navigation de robots ou l'aide à la conduite par exemple.

L'algorithme de base met en jeu un modèle qui décrit l'évolution temporelle de l'état du système et un modèle qui relie cet état à la mesure que l'on peut en faire. Le filtrage se décompose en trois étapes :

1. *L'étape de prédiction*, qui utilise le modèle d'état pour estimer la valeur prévue du paramètre, ainsi que la variance (ou matrice de covariance) associée.
2. *L'étape d'observation*, qui recueille une mesure.
3. *L'étape d'estimation*, qui pondère la mesure effectuée et la prédiction pour mettre à jour la variable d'état et la variance associée.

L'utilisation de ce filtre se limite aux cas où les modèles d'état et les mesures sont linéaires, et où les modèles de bruit sont additifs gaussiens. En effet, ces conditions remplies, la loi *a posteriori* présente l'avantage d'être une gaussienne et peut être entièrement définie par sa moyenne et sa covariance.

Dans le cas où les modèles d'état ne sont plus linéaires ou que les modèles de bruit ne sont plus gaussiens, il existe des solutions au problème : ce sont des extensions du filtre de Kalman. Parmi les plus courantes, on trouve le filtre de Kalman linéarisé et le filtre de Kalman étendu. Nous reviendrons en détails sur ces filtres au chapitre 3.

Par contre, lorsque les modèles d'état ne sont plus linéaires et que les modèles de bruit ne sont plus gaussiens, il faut se placer dans un cadre bayésien et utiliser des méthodes probabilistes pour résoudre le problème : c'est l'objet du paragraphe suivant.

Les approches probabilistes

Lorsque les modèles d'état ne sont plus linéaires et que les modèles de bruit ne sont plus gussiens, ce qui est souvent le cas en réalité, les calculs sont beaucoup plus complexes, et dès que la dimension de l'espace d'état est supérieure à quatre, les approximations ne peuvent plus être calculées.

Les méthodes probabilistes fournissent alors un cadre permettant d'estimer la loi *a posteriori* de manière simple. Les méthodes séquentielles de Monte Carlo proposent une approximation de cette loi par une somme de lois de Dirac centrées en des points appelés *particules*, et pondérées par la vraisemblance des mesures conditionnellement à ces points. Ces méthodes reposent sur le principe de Monte Carlo qui consiste à simuler un grand nombre de réalisations d'une variable aléatoire pour en approcher sa loi. Le concept est apparu au milieu du vingtième siècle mais n'a été mis en œuvre que beaucoup plus tard dans le cadre du problème de l'estimation séquentielle, car il a fallu attendre que la puissance de calcul des ordinateurs soit suffisante.

C'est dans ce cadre que se placent nos travaux de thèse ; nous expliciterons ces approches probabilistes en détail au chapitre 3.

2.5 Etat de l'art : détails de quelques approches

Jusqu'ici nous avons présenté les informations qui peuvent être extraites des images, la façon dont on peut les mettre en correspondance d'une image à l'autre d'une séquence, et les problèmes auxquels il fallait pouvoir faire face pour assurer un suivi robuste (changements d'illumination, présence d'ombres, occultations ...). Ces choix dépendent de l'information *a priori* que l'on a de l'objet que l'on veut suivre et de l'application finale. Ce paragraphe a pour but de faire le parallèle, pour quelques approches que nous allons décrire, entre les choix algorithmiques qui sont faits et les notions qui ont été présentées dans l'ensemble du chapitre.

Koller, Daniilidis et Nagel utilisent un modèle polyédrique pour représenter un modèle générique de véhicule [KDN94]. L'application est le suivi de véhicules sur des routes avec une unique caméra. L'information extraite dans les images est une information de contours : les segments rectilignes. Puis ces segments sont mis en correspondance avec ceux du modèle polyédrique reprojeté dans l'image courante. La distance de Mahalanobis est utilisée pour cette mise en correspondance. D'autre part, un algorithme permet de ne reprojeter que les arêtes visibles du modèle polyhédrique dans l'image. Un modèle dynamique de mouvement permet d'estimer la position du véhicule à l'instant suivant.

Bascle et Deriche combinent des informations de contour et de région pour assurer un suivi robuste d'objets déformables dans une séquence [BD95]. Un modèle de mouvement (rigide, affine ou homographique) permet d'estimer l'état de l'objet d'une image à l'autre. Une corrélation des niveaux de gris entre la région estimée et la région de référence permet de corriger le contour de la région déformable. La combinaison des informations de contour et de région permet d'améliorer les performances du suivi, particulièrement lors de la présence de textures sur l'objet suivi.

Pérez et Hue proposent dans [PHVG02] une méthode de suivi utilisant la couleur. L'objet est une ellipse dans laquelle un histogramme de couleur est calculé. Une méthode probabiliste (méthode de Monte Carlo, voir au chapitre 3 pour plus de détails) permet d'estimer l'état de l'objet à l'instant suivant, et la distance de Bhattacharyya est utilisée pour mesurer la similarité entre deux histogrammes. Se placer dans un cadre probabiliste permet une grande flexibilité dans la modélisation de l'objet, du bruit, et de tout autre modèle (illumination, mouvement, ...).

Nummiaro et al. [NKMG03] proposent eux aussi un algorithme de suivi d'objets combinant approche probabiliste et utilisation de la couleur pour modéliser l'objet d'intérêt. De la même

manière que dans l'approche de Pérez et Hue, une distribution de couleur est calculée dans une forme géométrique servant à modéliser l'objet d'intérêt. Cependant les deux approches diffèrent sur les points suivants :

- la forme de la boîte englobante : une grande variété de forme pour Pérez et Hue, une ellipse dont les pixels sont pondérés par leur distance au centre de l'ellipse pour Nummiaro et al.
- l'espace des couleurs pour le calcul des histogrammes : HSV (Hue Saturation Value) dans le cas de Pérez et Hue, RVB dans l'approche de Nummiaro
- la dynamique de mouvement : de type $x_{t+1} = Ax_t + Bx_{t-1} + Cv_t$ où $v_t \sim N(O, \Sigma)$ (basé sur l'apprentissage de séquences types) pour Perez et Hue, 2^{nd} ordre (c'est-à-dire vitesse constante, $x_t = Ax_{t-1} + w_{t-1}$ avec w_{t-1} variable aléatoire gaussienne) pour l'autre approche
- la mise à jour du modèle : inexistante pour Pérez et Hue, adaptative dans l'approche de Nummiaro et al.
- l'initialisation : manuelle dans la première image pour Pérez et Hue, elle est semi-automatique dans l'approche de Nummiaro et al.

L'algorithme a été testé pour du suivi de personnes et de voitures dans le cadre de vidéo surveillance ; les séquences de tests sont celles de PETS 2001 (Performance Evaluation of Tracking Systems). La gestion des occultations se fait par détection : on détecte les endroits où l'objet est suceptible de réapparaître, puis on calcule des mesures de similarité à ces endroits là. Cela permet à l'algorithme de rester performant en cas d'occultations courtes, partielles ou totales. Des résultats sont montrés sur une séquence où une personne marche et passe derrière un arbre.

Dans les approches présentées ci-dessus, les informations extraites des images sont simples, et il n'y a pas d'étapes de « fusion d'informations » pour combiner couleurs, formes, contours ou autres informations. Pérez, Vermaak et Blake se sont penchés sur le problème. L'algorithme qu'ils proposent dans [PP04] est un filtre particulaire combinant des informations de trois types : couleur, son et mouvement. La couleur est considérée comme l'information principale ; elle est combinée, selon le scénario considéré, à des informations de son ou de mouvement. L'information de couleur utilisée est un histogramme calculé dans l'espace RVB, que l'on compare à un histogramme de référence qui sert de modèle. Pour des applications de suivi audio-visuel, une caméra est combinée à une paire de microphones, et l'information de son est mesurée par le temps que met le son à arriver aux deux microphones. Pour les autres applications, on combine des informations de couleur et de mouvement. L'information de mouvement est obtenue par différence d'images successives. Les auteurs comparent sur différentes séquences les résultats obtenus en utilisant un seul type d'information, puis ceux obtenus en utilisant la fusion d'information. Les résultats présentés montrent l'intérêt de combiner des informations de types différents pour améliorer les performances du suivi d'objet.

Les méthodes probabilistes sont aussi utilisées pour d'autres applications de suivi. Par exemple, dans le cadre d'une application militaire, on cherche à suivre une cible à partir de mesures d'angles. Le pistage de cible par mesure d'angle consiste à déterminer la trajectoire d'une cible en utilisant des mesures d'angles bruitées obtenues par un observateur. La fonction d'observation étant non linéaire, les approches probabilistes, et notamment le filtrage particulaire, sont très adaptées pour la résolution du problème. Bréhard et Le Cadre proposent un algorithme de pistage mono-cible par mesure d'angle seul [BC04].

Yokoyama et Poggio proposent un algorithme rapide et robuste pour la détection et le suivi d'objets [YP05]. Leur méthode repose sur l'utilisation de lignes dans les images. L'extraction des

contours se fait en quatre étapes : le détecteur de contour de Canny est utilisé pour extraire des contours dans l'image. Ces contours pouvant être fragmentés, ils utilisent alors un détecteur de lignes pour restaurer les lignes des contours extraits. Les contours obtenus sont ceux des objets, mais aussi ceux du fond. Pour éliminer les contours appartenant au fond (et ne conserver que ceux des objets en mouvement), on soustrait les contours obtenus aux lignes du fond de l'image précédente. Une étape de clustering permet d'attribuer un label pour chaque ligne, le label étant le même pour deux lignes appartenant au même objet. Enfin les contours de ces lignes groupées sont extraits en utilisant des contours actifs. La phase de suivi se déroule comme suit : les objets détectés sont ensuite suivis, la mesure de ressemblance utilisée pour comparer un objet de l'image précédente et un objet de l'image courante est basée sur les positions des lignes connues de l'image précédente et estimées en utilisant le flot optique pour l'image courante. Chacun des objets a un état qui permet de gérer les occultations ou les interférences avec les autres objets. Les expérimentations sont effectuées sur des séquences prises en extérieur pour le suivi de personnes ; les résultats présentés montrent les performances de l'algorithme, notamment en termes de robustesse et de rapidité.

La dernière approche que nous allons présenter est celle de Comaniciu [CRM, CRM03]. Il propose une méthode de suivi en utilisant la proédure mean shift. A l'origine, la procédure mean shift est une procédure itérative de montée de gradient utilisée pour estimer les modes d'une densité assoiée à échantillon d'observations. Elle a été proposée par Fukunaga en 1975 et a été utilisée pour la première fois en 1997 dans le cadre de la segmentation d'images, puis a été adaptée en tant que méthode de suivi par Comaniciu et al. L'algorithme de suivi qu'il propose consiste en la poursuite d'un objet dans une séquence d'images, à partir de sa position dans la première image : l'initialisation est donc manuelle. L'objet d'intérêt est modélisé par une ellipse, dans laquelle on calcule sa distribution de couleur. La distribution de couleur initiale est référencée en tant que modèle, et est ensuite comparée à celles des sites candidats pour déterminer la position la plus probable dans l'image suivante.

Le mode d'un nuage de point correspond à un maximum local de sa densité, ce qui implique qu'en ce point le gradient soit égal au vecteur nul. Le principe de la procédure mean shift est de trouver le mode en résolvant itérativement l'équation $\bigtriangledown f(x) = 0$ sans estimer la densité f. En pratique, on ne connaît pas la densité réelle du nuage ; on ne peut donc pas calculer directement son gradient. L'astuce de Fukunaga est d'estimer le gradient de la fonction de densité avec le gradient de l'estimation de la foncion de densié. Nous n'allons pas rentrer dans les détails de la procédure, nous vous invitons à vous référer à l'article de Comaniciu [CRM, CRM03] pour plus de détails.

L'algorithme qu'il propose fonctionne de la manière suivante : pour chaque image, il utilise la position de la cible estimée à l'image précédente comme initialisation. Il calcule la distribution de couleur dans cette ellipse, et évalue la similarité avec la distribution du modèle. Un poids est associé à cette mesure de similarité et et on peut alors calculer le vecteur mean shift, qui a pour but de fournir la nouvelle position estimée de la cible dans l'image courante. On réitère ces étapes jusqu'à trouver le maximum local.

Les tests ont été effectués dans le cadre du suivi de joueurs de football dans une vidéo. L'algorithme tourne en temps réel, et les résultats présentés dans l'article montrent l'intérêt de l'utilisation d'histogrammes et de l'utilisation du vecteur mean shift pour faire du suivi temps réel. Notons quand même que l'algorithme fonctionne bien si deux objets présentant les mêmes distributions de couleur ne sont pas trop proches dans les images.

Chapitre 3

Le filtrage

Sommaire

3.1 Introduction

3.1.1 La problématique

Dans notre contexte, le filtrage consiste à estimer l'état d'un système dynamique, c'est-à-dire qui évolue au cours du temps, à partir d'observations. Ces observations sont partielles, généralement bruitées et parfois incomplètes. Pour cela, on dispose d'une suite d'observations, que l'on notera par la suite $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n$. Chaque observation $\mathbf{Y}_n$ est reliée à l'état inconnu $\mathbf{X}_n$ par une relation du type :

$$\mathbf{Y}_n = h(\mathbf{X}_n) + \mathbf{W}_n$$

où $\mathbf{W}_n$ est un bruit qui modélise l'erreur d'observation.

Tel qu'il est formulé, le problème d'estimation de l'état inconnu $\mathbf{X}_n$ à partir des observations $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n$ est en général mal posé. Il existe la plupart du temps une infinité de solutions. On peut se reporter au cours de Legland pour une démonstration détaillée [Leg05]. Pour lever

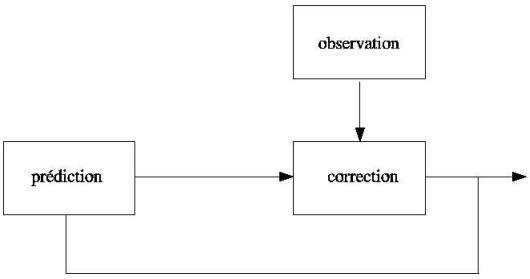

FIG. 3.1 – Etapes du filtrage : La prédiction fournit l'estimé de l'état à l'image suivante, et la correction ajuste la valeur de l'état actuel en tenant compte de la prédiction et de l'observation effectuée.

l'indétermination, on essaie d'utiliser de l'information supplémentaire, sur le paramètre inconnu par exemple. L'information supplémentaire rajoutée est appelée *information a priori* sur l'état inconnu. On peut également estimer une distribution et non pas une valeur ; dans ce cas, l'indétermination n'est pas levée mais elle est tout de même prise en compte.

Quelques soient les hypothèses faites sur le modèle d'état, le système est décrit par trois éléments :
- La distribution du processus d'état à l'instant initial ; on le notera par la suite $\mathbf{X}_0$
- Un modèle d'évolution de l'état, fixé *a priori*
- Un modèle de vraisemblance qui relie l'état à l'observation

Selon le type d'application, on distingue trois problèmes : le problème du *filtrage* consiste à rechercher la distribution de l'état à l'instant courant connaissant l'ensemble des observations passées et présentes. En raison de la complexité calculatoire, on n'utilise explicitement que l'observation précédente ; cependant, pour l'estimation de l'état, toutes les observations passées sont utilisées (mais implicitement). Le problème du *lissage* utilise les mesures « futures » pour estimer la distribution de l'état ; dans ce cas, il n'y a plus de notion de temps. Enfin, le problème de *prédiction* consiste à rechercher la loi de l'état futur à partir des observations passées et présentes. Notre application est celle du suivi dans une séquence d'images. L'algorithme de suivi doit pouvoir fonctionner en temps réel : il est donc indispensable que les coûts de calcul soient faibles. Pour répondre à ces besoins, nous nous limiterons au problème du *filtrage*, et utiliserons un filtre récursif qui permet de calculer la probabilité de l'état *a posteriori*.

Le schéma général d'un algorithme récursif se décompose en deux étapes :

1. L'étape de *prédiction* utilise un modèle d'évolution *a priori* afin d'approcher la distribution de l'état futur conditionnellement aux mesures disponibles
2. L'étape de *correction* permet de mettre à jour la distribution prédite à l'aide de la nouvelle observation et du modèle de vraisemblance associé.

Ces deux étapes sont résumées sur le schéma de la figure 3.1.

Selon le type de modèle d'évolution et de vraisemblance (linéaires ou pas, gaussiens ou pas), différents algorithmes ont été proposés pour résoudre le problème de filtrage. Ainsi, pour des

modèles linéaires et gaussiens, le filtre de Kalman (dans le cas discret) ou celui de Kalman-Bucy (dans le cas continu) fournit la solution optimale au problème. Dès que les modèles ne sont plus linéaires, les approches probabilistes fournissent une solution. Nous allons, après un rapide état de l'art, détailler quelques-unes de ces approches.

3.1.2 Etat de l'art

Au début des années 1940, dans le cadre de recherches militaires, Wiener mit au point le premier filtre, couramment appelé « Filtre de Wiener ». En 1961, Kalman et Bucy ont introduit le filtre de Kalman [Kal60], qui enrichit le filtre de Wiener mais qui n'est développé que pour des modèles linéaires gaussiens ; c'est un filtre récursif et qui peut être appliqué à des systèmes décrits par des modèles variant avec le temps. Cependant, les hypothèses de modèles linéaires gaussiens ne sont pas toujours vérifiées en pratique. Des solutions alternatives ont été développées (le filtre de Kalman étendu par exemple, qui linéarise le modèle dynamique autour d'une solution approchée et qui est utilisé dans la plupart des cas non linéaires, ou encore des filtres utilisant des HMM (Hidden Markov Models) [RJ86]) mais n'assurent pas une convergence de l'algorithme. D'autre part, en plus des non linéarités le problème du filtrage est souvent perturbé par la présence de mesures non issues des objets (appelées *fausses alarmes*), et qui conduisent à la multi modalité de la loi conditionnelle de l'état. Dans ce cas, les solutions proposées ci-dessus sont inadaptées et ne peuvent être utilisées [AM79].

Dans le cas général, les non linéarités des modèles, la complexité des dynamiques, et la gestion des fausses alarmes sont des obstacles qui ne permettent pas de résoudre le problème du filtrage de manière optimale. La densité conditionnelle est décrite par des équations différentielles, et son évolution fait intervenir des intégrales multi dimensionnelles. Pour évaluer cette densité, des méthodes ont été développées mais lorsque la dimension de l'état est supérieure à trois, ces méthodes sont extrêmement lourdes à calculer. Ces approches reposent sur la discrétisation de l'espace d'état sous forme de maillage [Kit87, Sor88]. Elles permettent d'estimer une approximation numérique de la loi *a posteriori* pour les chaînes de Markov cachées définies sur des espaces continus, mais nécessitent des temps de calculs extrêmement élevés dès que la dimension de l'espace d'état dépasse quatre pour que les approximations soient correctes.

C'est au début des années 1990 que les méthodes de Monte Carlo [Kit96] sont proposées pour résoudre le filtrage non linéaire. Contrairement aux algorithmes évoqués jusqu'à maintenant, les méthodes séquentielles de Monte Carlo sont des algorithmes stochastiques, reposant sur le principe de Monte Carlo qui consiste à simuler un grand nombre de réalisations d'une variable aléatoire pour en approcher sa loi. Le concept est apparu dès le milieu du vingtième siècle dans le domaine de la physique, mais il a fallu attendre l'augmentation de la puissance de calcul des ordinateurs pour pouvoir les mettre en œuvre dans le cadre de l'estimation séquentielle et du filtrage non linéaire. Ces méthodes ont des performances peu sensibles à la dimension de l'espace d'état. Cela les rend particulièrement intéressantes puisqu'elles permettent d'estimer la densité conditionnelle en temps réel. Les méthodes particulaires ont été introduites par Gordon, Salmond et Smith [GSS93], et sont une version séquentielle des méthodes de Monte Carlo pour résoudre le problème du filtrage. Nous allons décrire ces méthodes dans la suite du chapitre.

3.2 Les méthodes non aléatoires : le filtrage de Kalman

Comme on l'a dit en introduction, si l'on se place dans un cadre gaussien, l'évolution de la loi conditionnelle est régie par un système dynamique, le filtre de Kalman-Bucy, simple à mettre en œuvre. Dans tous les autres cas, l'évolution de cette loi conditionnelle est déterminée par un autre type de système, souvent impossible à utiliser en pratique. Cependant, les techniques développées dans le cas linéaire peuvent s'étendre au cas non linéaire par des méthodes de linéarisation. Les filtres obtenus sont le plus souvent utilisables en pratique mais conduisent parfois à de mauvais résultats.

Dans cette section, nous détaillons le filtre de Kalman et deux de ses dérivées : le filtre de Kalman linéarisé et le filtre de Kalman étendu. Ces deux filtres dérivés sont les plus utilisés en pratique.

3.2.1 Notations

Avant de présenter le filtre de Kalman-Bucy, il est nécéssaire d'introduire quelques définitions et notations. Dans toute la suite du chapitre, on considérera l'équation d'état suivante :

$$\mathbf{X}_k = F_k \mathbf{X}_{k-1} + f_k + \mathbf{W}_k$$

avec les hypothèses :
- k représente les instants successifs du temps
- $\mathbf{X}_k$ est un vecteur représentant l'état du système, à valeurs dans $\mathbb{R}^m$. Le vecteur représentant l'état initial $\mathbf{X}_0$ est de composantes gaussiennes, d'espérance (ou moyenne) $\bar{X}_0$ et de covariance Q_0^X
- F_k est déterministe, appelée matrice d'état, à valeurs dans $\mathbb{R}^{m\times m}$
- f_k est une entrée connue du système
- $\mathbf{W}_k$ est un bruit blanc gaussien, à valeurs dans $\mathbb{R}^m$, de covariance Q_k^W. On suppose que $\mathbf{W}_k$ est indépendant de la condition initiale $\mathbf{X}_0$.

On considérera de même l'équation d'observation suivante :

$$\mathbf{Y}_k = H_k \mathbf{X}_k + h_k + \mathbf{V}_k$$

avec les hypothèses :
- k représente les instants successifs du temps
- $\mathbf{Y}_k$ est la sortie du système (mesure ou observation), à valeurs dans $\mathbb{R}^d$
- H_k est déterministe, appelée matrice d'observation, à valeurs dans $\mathbb{R}^{d\times m}$
- h_k est une entrée connue du système
- $\mathbf{V}_k$ est un bruit blanc gaussien, à valeurs dans $\mathbb{R}^d$, de covariance Q_k^V. On suppose que $\mathbf{V}_k$ est indépendant de la condition initiale $\mathbf{X}_0$, et du bruit $\mathbf{W}_k$.

En résumé : $\mathbf{X}_k$ représente l'état d'un système à l'instant k. On suppose que l'on ne peut pas observer directement ce système, mais que l'on dispose d'une observation $\mathbf{Y}_k$ qui est la somme d'un signal $H_k\mathbf{X}_k + h_k$ et d'un bruit d'observation $\mathbf{V}_k$. On cherche à estimer l'état du sytème.

3.2.2 Le filtre de Kalman

On considère un processus $\{\mathbf{X}_k\}$ représentant l'état d'un système non observé. A l'instant k, on recueille une observation $\mathbf{Y}_k$ qui est formée d'une fonction $h(\mathbf{X}_k)$ de l'état $\mathbf{X}_k$ et d'un bruit

additif $\mathbf{V}_k$.

Considérons le système linéaire à bruit additif suivant :

$$\left\{ \begin{array}{rcl} \mathbf{X}_k & = & F_k \mathbf{X}_{k-1} + f_k + \mathbf{W}_k \\ \mathbf{Y}_k & = & H_k \mathbf{X}_k + h_k + \mathbf{V}_k \end{array} \right. \tag{3.1}$$

avec les hypothèses du paragraphe 3.2.1.

A l'instant k, on dispose de l'information $\mathbf{Y}_{0:k} = (\mathbf{Y}_0, \mathbf{Y}_1, \ldots, \mathbf{Y}_k)$. L'objectif est d'estimer le vecteur $\mathbf{X}_k$ à partir de $\mathbf{Y}_{0:k}$ de façon optimale et récursive. Il s'agit donc de calculer la loi conditionnelle du vecteur aléatoire $\mathbf{X}_k$ sachant $\mathbf{Y}_{0:k}$.

En notant $\mathcal{N}(\mathbf{X}; \bar{X}, Q^X)$ la loi normale de variable $\mathbf{X}$, de moyenne $\bar{X}$ et de covariance Q^X, on peut écrire le système de la façon suivante :

$$\left\{ \begin{array}{rcl} \mathbf{X}_0 & \sim & \mathcal{N}(\mathbf{X}_0; \bar{X}_0, Q_0^X) \\ \mathbf{X}_k | \mathbf{X}_{k-1} & \sim & \mathcal{N}(\mathbf{X}_k; F_k \mathbf{X}_{k-1} + f_k, Q_k^W) \\ \mathbf{Y}_k | \mathbf{X}_k & \sim & \mathcal{N}(\mathbf{Y}_k; H_k \mathbf{X}_k + h_k, Q_k^V) \end{array} \right. \tag{3.2}$$

Par linéarité des équations d'état et de mesure, le processus $\{\mathbf{X}_k, \mathbf{Y}_k\}$ est gaussien et la loi de filtrage recherchée $p(\mathbf{X}_k | \mathbf{Y}_{0:k})$ est gaussienne. Par conséquent, le problème est de dimension finie, et la loi de filtrage est entièrement décrite par son espérance et sa covariance qui s'écrivent :

$$\hat{\mathbf{X}}_{k|k} = \mathbb{E}\left[\mathbf{X}_k | \mathbf{Y}_{0:k}\right] \qquad \text{et} \qquad P_{k|k} = \mathbb{E}\left[(\mathbf{X}_k - \hat{\mathbf{X}}_{k|k})(\mathbf{X}_k - \hat{\mathbf{X}}_{k|k})^T\right]$$

Supposons que la loi conditionnelle du vecteur aléatoire $\mathbf{X}_{k-1}$ sachant $\mathbf{Y}_{0:k-1}$ soit connue. Pour calculer la loi conditionnelle du vecteur aléatoire $\mathbf{X}_k$ sachant $\mathbf{Y}_{0:k}$, on procède en deux étapes :

1. L'étape de *prédiction* permet de calculer la loi conditionnelle du vecteur aléatoire $\mathbf{X}_k$ connaissant les observations passées $\mathbf{Y}_{0:k-1}$. On utilise pour cela l'équation d'état du système (3.1).
2. L'étape de *correction* utilise la nouvelle observation $\mathbf{Y}_k$. Plus précisément, on prend en compte la composante de l'observation $\mathbf{Y}_k$ qui apporte une information nouvelle par rapport aux observations passées $\mathbf{Y}_{0:k-1}$, c'est-à-dire

$$I_k = \mathbf{Y}_k - \mathbb{E}\left[\mathbf{Y}_k | \mathbf{Y}_{0:k-1}\right]$$

Le filtre de Kalman-Bucy fournit la solution au problème. En supposant que la matrice de covariance Q_k^V est inversible pour tout $k \in \mathbb{N}$, alors $\{\hat{X}_k\}$ et $\{P_k\}$ sont définis par les équations suivantes :

$$\left\{ \begin{array}{rcl} \hat{\mathbf{X}}_{k|k-1} & = & F_k \hat{\mathbf{X}}_{k-1} + f_k \\ P_{k|k-1} & = & F_k P_{k-1|k-1} F_k^T + Q_k^W \end{array} \right. \tag{3.3}$$

et

$$\left\{ \begin{array}{rcl} \hat{\mathbf{X}}_{k|k} & = & \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - (H_k \hat{\mathbf{X}}_{k|k-1} + h_k)\right] \\ P_{k|k} & = & \left[I - K_k H_k\right] P_{k|k-1} \end{array} \right. \tag{3.4}$$

où la matrice K_k, définie par

$$K_k = P_{k|k-1} H_k^T \left[H_k P_{k|k-1} H_k^T + Q_k^V \right]^{-1}$$

est appelée *gain de Kalman.*

L'initialisation du filtre est donnée par :

$$\left\{ \begin{array}{rcccl} \hat{\mathbf{X}}_{0|0} & = & \bar{X}_0 & = & \mathbb{E}\left[\mathbf{X}_0\right] \\ P_{0|0} & = & Q_0^X & = & cov(X_0) \end{array} \right.$$

Remarquons que les matrices de covariance et le gain du filtre ne dépendent pas de la mesure courante. Par conséquent on pourra calculer hors ligne ces matrices dans le cas où les matrices du système F_k, H_k, Q_k^W et Q_k^V sont indépendantes du temps.

Concernant le fonctionnement du filtre de Kalman : durant la phase de correction, le gain de Kalman est multiplié par un terme que l'on appelle *innovation* et qui permet de corriger l'estimée prédite. Elle est définie par l'expression $(H_k \hat{X}_{k|k-1} + h_k)$, ce qui correspond à la différence entre la mesure observée et la mesure prédite. L'interprétation de cette étape est la suivante : une grande confiance dans les estimations précédentes et un doute dans les mesures impliquent un gain faible. La prédiction est alors faiblement corrigée. A l'inverse, un doute sur les estimations précédentes et une confiance en la nouvelle mesure impliquent un gain fort. Et la mesure prend davantage d'importance dans la valeur finale de l'estimée.

– Initialisation :

$$\hat{\mathbf{X}}_{0|0} = \bar{X}_0 = \mathbb{E}\left[\mathbf{X}_0\right]$$

$$P_{0|0} = Q_0^X = cov(\mathbf{X}_0)$$

– Pour tout $k \in \mathbb{N}, \quad k \geq 1$

• Prédiction :

$$\hat{\mathbf{X}}_{k|k-1} = F_k \hat{\mathbf{X}}_{k-1|k-1} + f_k$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k^W$$

• Correction :

$$K_k = P_{k|k-1} H_k^T \left[H_k P_{k|k-1} H_k^T + Q_k^V \right]^{-1}$$

$$\hat{\mathbf{X}}_{k|k} = \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - (H_k \hat{\mathbf{X}}_{k|k-1} + h_k) \right]$$

$$P_{k|k} = \left[I - K_k H_k \right] P_{k|k-1}$$

FIG. 3.2 – Le filtre de Kalman.

Pour calculer les équations du filtre de Kalman, deux approches sont possibles. La première consiste à calculer explicitement les deux étapes du filtre optimal, ce qui est possible grâce aux propriétés des lois gaussiennes. Cependant, ce premier type d'approche n'est plus valable lorsqu'on sort du cadre gaussien. La seconde approche repose sur la théorie de l'estimation dans

un cadre beaucoup plus général. Dans ce cadre, l'estimateur de variance minimale est défini par $\hat{\mathbf{X}}_{k|k} = \mathbb{E}[\mathbf{X}_k - \mathbf{Y}_{0:k}]$. L'hypothèse gaussienne permet de simplifier l'expression précédente, l'estimateur étant alors équivalent à un estimateur linéaire non biaisé de variance minimale. Le problème se ramène donc à un problème d'estimation aux moindres carrés. La procédure est la suivante : sous la contrainte que l'estimateur est une combinaison linéaire de l'innovation et de l'estimateur prédit, on recherche le gain qui minimise la covariance de l'erreur d'estimation. On peut se référer à l'ouvrage de Gelb [Gel74] pour plus de détails.

L'algorithme du filtre de Kalman est représenté sur la figure 3.2.

Une dernière remarque concerne les processus de bruit. Dans le cas où ils ne sont plus supposés gaussiens, mais que l'on dispose de leurs deux premiers moments (moyenne et covariance), le filtre de Kalman fournit le meilleur des estimateurs linéaires au sens où il minimise la covariance de l'erreur parmi les estimateurs linéaires. Dans le cas où les modèles de bruits sont gaussiens mais ne sont plus supposés indépendants entre eux et indépendants de la condition initiale, il est possible de se ramener au cas d'un modèle à bruits décorrélés par la construction d'un modèle équivalent. Un nouveau vecteur de bruit est défini, et on peut dériver le nouveau filtre à partir du filtre classique. On retrouve les étapes de prédiction et de correction du filtre classique, qui ont été modifiées par ajout d'un terme correcteur. On peut se référer aux ouvrages de Gelb [Gel74] ou Anderson [AM79] pour une présentation détaillée du filtre de Kalman et de ses dérivées.

3.2.3 Filtre de Kalman linéarisé

Le filtre de Kalman tel qu'il a été présenté à la section précédente permet d'estimer dans le temps l'état d'un procédé défini par une équation linéaire. Cependant, dans la réalité, l'hypothèse de linéarité d'un procédé n'est pas toujours vérifiée. Le filtre de Kalman ne peut donc pas être utilisé, mais cela ne veut pas dire que l'on ne peut rien réutiliser de ce qui a été présenté précédemment.

En effet, une astuce permet d'ajuster le filtre pour un procédé avec une équation non linéaire pour linéariser les variables dont le filtre a besoin. L'idée est de linéariser localement sur la moyenne et la variance de la sortie du procédé. Avec les séries de Taylor, une linéarisation de l'estimation peut être effectuée en utilisant les dérivées partielles des fonctions du procédé F et de mesure H. Cette linéarisation utilise l'hypothèse que l'erreur sur la dérivée est petite ; ainsi, on peut tronquer la série de Taylor dès le premier ordre.

En reprenant les notations de la section précédente, on considère maintenant le système non linéaire suivant :

$$\left\{ \begin{array}{rcl} \mathbf{X}_k & = & f_k(\mathbf{X}_{k-1}) + \mathbf{W}_k \\ \mathbf{Y}_k & = & h_k(\mathbf{X}_k) + \mathbf{V}_k \end{array} \right. \tag{3.5}$$

avec les hypothèses suivantes :

- les fonctions f_k et h_k sont définies sur $\mathbb{R}^m$, à valeurs dans $\mathbb{R}^m$ et $\mathbb{R}^{m \times p}$ respectivement. De plus, on suppose que h_k est dérivable.
- $\{\mathbf{W}_k\}$ et $\{\mathbf{V}_k\}$ sont des bruits blancs gaussiens (de covariances respectives Q_k^W et Q_k^V), indépendants entre eux, et indépendants de la condition initiale $\mathbf{X}_0$.

On se donne $\{\hat{\mathbf{x}}_k\}$ une suite déterministe dans $\mathbb{R}^m$, appelée *trajectoire nominale*. Un cas courant est de prendre $\hat{\mathbf{x}}_k$ comme une approximation de la moyenne de $\mathbf{X}_k$. La méthode consiste à linéariser la fonction f_k autour de $\hat{\mathbf{x}}_{k-1}$, c'est-à-dire

$$f_k(x) \simeq f_k(\hat{\mathbf{x}}_{k-1}) + f'_k(\hat{\mathbf{x}}_{k-1})(\mathbf{x} - \hat{\mathbf{x}}_{k-1})$$

et la fontion h_k autour de $\hat{\mathbf{x}}_k$, soit

$$h_k(x) \simeq h_k(\hat{\mathbf{x}}_k) + h'_k(\hat{\mathbf{x}}_k)(\mathbf{x} - \hat{\mathbf{x}}_k)$$

Le système (3.5) est alors remplacé par le système

$$\left\{ \begin{array}{rcl} \mathbf{X}_k &=& F_k(\mathbf{X}_{k-1} - \hat{\mathbf{x}}_{k-1}) + f_k + \mathbf{W}_k \\ \mathbf{Y}_k &=& H_k(\mathbf{X}_k - \hat{\mathbf{x}}_k) + h_k + \mathbf{V}_k \end{array} \right. \tag{3.6}$$

avec $F_k = f'_k(\hat{\mathbf{x}}_{k-1})$, $f_k = f_k(\hat{\mathbf{x}}_{k-1})$, $H_k = h'_k(\hat{\mathbf{x}}_k)$ et $h_k = h_k(\hat{\mathbf{x}}_k)$. On applique alors le filtre de Kalman à ce nouveau système, et on obtient exactement

$$\begin{array}{rcl} \hat{\mathbf{X}}_{k|k-1} &=& f_k(\hat{\mathbf{x}}_{k-1}) + f'_k(\hat{\mathbf{x}}_{k-1})(\hat{\mathbf{X}}_{k-1} - \hat{\mathbf{x}}_{k-1}) \\ P_{k|k-1} &=& f'_k(\hat{\mathbf{x}}_{k-1}) P_{k-1|k-1} \left[f'_k(\hat{\mathbf{x}}_{k-1})\right]^T + g_k(\hat{\mathbf{x}}_{k-1}) Q_k^W \left[g_k(\hat{\mathbf{x}}_{k-1})\right]^T \\ \\ \hat{\mathbf{X}}_{k|k} &=& \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - \left[h'_k(\hat{\mathbf{x}}_k)(\hat{\mathbf{X}}_{k|k-1} - \hat{\mathbf{x}}_k) + h_k(\hat{\mathbf{x}}_k)\right]\right] \\ P_{k|k} &=& \left[I - K_k h'_k(\hat{\mathbf{x}}_k)\right] P_{k|k-1} \\ K_k &=& P_{k|k-1} h'_k(\hat{\mathbf{x}}_k)^T \left[h'_k(\hat{\mathbf{x}}_k) P_{k|k-1} \left[h'_k(\hat{\mathbf{x}}_k)\right] + Q_k^V\right]^{-1} \end{array}$$

A la place de la première et de la troisième des équations écrites ci-dessus, on peut utiliser :

$$\begin{array}{rcl} \hat{\mathbf{X}}_{k|k-1} &=& f_k(\hat{\mathbf{X}}_{k-1|k-1}) \\ \hat{\mathbf{X}}_{k|k} &=& \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - h_k(\hat{\mathbf{X}}_{k|k-1})\right] \end{array}$$

Enfin, on choisit l'initialisation $\hat{\mathbf{X}}_{0|0}$ et Q_0^V de façon à ce que $\mathcal{N}(\hat{\mathbf{X}}_{0|0}; \bar{X}_0, Q_0^X)$ soit une bonne approximation de la loi de $\mathbf{X}_0$.

3.2.4 Filtre de Kalman étendu

Les coefficients du système linéaire peuvent également dépendre des observations (jusqu'à l'instant $k-1$). Dans ce cas, au lieu d'utiliser une trajectoire nominale déterministe, on peut utiliser l'estimateur courant. La méthode consiste à linéariser la fonction f_k autour de $\hat{\mathbf{X}}_{k-1|k-1}$, c'est-à-dire

$$f_k(\mathbf{x}) \simeq f_k(\hat{\mathbf{X}}_{k-1|k-1}) + f'_k(\hat{\mathbf{X}}_{k-1|k-1})(\mathbf{x} - \hat{\mathbf{X}}_{k-1|k-1})$$

et à linéariser la fonction h_k autour de $\hat{\mathbf{X}}_{k|k-1}$, c'est-à-dire

$$h_k(\mathbf{x}) \simeq h_k(\hat{\mathbf{X}}_{k|k-1}) + h'_k(\hat{\mathbf{X}}_{k|k-1})(\mathbf{x} - \hat{\mathbf{X}}_{k|k-1})$$

Le système (3.5) est alors remplacé par

$$\left\{ \begin{array}{rcl} \mathbf{X}_k &=& F_k(\mathbf{X}_{k-1|k-1} - \hat{\mathbf{X}}_{k-1|k-1}) + f_k + \mathbf{W}_k \\ \mathbf{Y}_k &=& H_k(\mathbf{X}_{k|k} - \hat{\mathbf{X}}_{k|k-1}) + h_k + \mathbf{V}_k \end{array} \right. \tag{3.7}$$

avec $F_k = f_k'(\hat{\mathbf{X}}_{k-1|k-1})$, $f_k = f_k(\hat{\mathbf{X}}_{k-1|k-1})$, $H_k = h_k'(\hat{\mathbf{X}}_{k|k-1})$ et $h_k = h_k(\hat{\mathbf{X}}_{k|k-1})$. On applique alors le filtre de Kalman à ce nouveau système, et on obtient exactement le résultat suivant :

$$\begin{array}{rcl} \hat{\mathbf{X}}_{k|k-1} &=& f_k(\hat{\mathbf{X}}_{k-1|k-1}) \\ P_{k|k-1} &=& F_k P_{k-1|k-1} F_k^Y + Q_k^W \end{array}$$

$$\begin{array}{rcl} \hat{\mathbf{X}}_{k|k} &=& \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - \left[h_k(\hat{\mathbf{X}}_{k|k-1}) \right] \right] \\ P_{k|k} &=& \left[I - K_k H_k \right] P_{k|k-1} \\ K_k &=& P_{k|k-1} H_k^T \left[H_k P_{k|k-1} H_k^T + Q_k^V \right]^{-1} \end{array}$$

avec $F_k = f_k'(\hat{\mathbf{X}}_{k-1|k-1})$ et $H_k = h_k'(\hat{\mathbf{X}}_{k|k-1})$.

L'algorithme du filtre de Kalman étendu est représenté sur la figure 3.3.

- Initialisation :

$$\hat{\mathbf{X}}_{0|0} = \bar{X}_0 = \mathbb{E}\left[\mathbf{X}_0\right]$$

$$P_{0|0} = Q_0^X = cov(\mathbf{X}_0)$$

- Pour tout $k \in \mathbb{N}, \quad k \geq 1$
  - Prédiction :

$$\hat{\mathbf{X}}_{k|k-1} = f_k(\hat{\mathbf{X}}_{k-1|k-1})$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^Y + Q_k^W$$

$$\text{où} \quad F_k = f_k'(\hat{\mathbf{X}}_{k-1|k-1})$$

  - Correction :

$$K_k = P_{k|k-1} H_k^T \left[H_k P_{k|k-1} H_k^T + Q_k^V \right]^{-1}$$

$$\hat{\mathbf{X}}_{k|k} = \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - \left[h_k(\hat{\mathbf{X}}_{k|k-1}) \right] \right]$$

$$P_{k|k} = \left[I - K_k H_k \right] P_{k|k-1}$$

$$\text{où} \quad H_k = h_k'(\hat{\mathbf{X}}_{k|k-1})$$

FIG. 3.3 – Le filtre de Kalman étendu.

Le filtre de Kalman étendu repose donc sur une approximation de la loi de filtrage par une loi gaussienne. Il permet de calculer les deux premiers moments de cette loi (moyenne et variance)

par linéarisation. Lorsque les bruits d'observation sont faibles, des travaux démontrent que l'utilisation de ce filtre dans un cas général est valide [Pic91, dO94].

Lorsque les non linéarités sont trop importantes, le filtre de Kalman étendu diverge. Pour remédier à ce problème, il a été proposé de linéariser les fonctions à des ordres plus importants, ou d'augmenter la variance des bruits pour prendre en compte l'erreur de linéarisation. D'autre part, pour améliorer la linéarisation, certains ont intégré un procédé itératif : la mesure courante est utilisée pour mettre à jour l'estimée autour de laquelle on peut réécrire un développement de Taylor pour appliquer à nouveau un filtre de Kalman. C'est le filtre de Kalman étendu itéré.

3.2.5 Le filtre de Kalman « unscented »

Nous présentons dans cette section une dernière version du filtre de Kalman, appelée *filtre de Kalman unscented*, et dénotée en anglais UKF. Ce filtre a été proposé par Julier en 1997 [JU97] pour éviter la linéarisation effectuée dans les filtres de Kalman étendus.

L'idée de Julier est d'approximer la loi conditionnelle *a posteriori* par une loi gaussienne de moyenne $\bar{X}_{k|k}$ et de covariance $P_{k|k}$. La loi gaussienne est définie par un ensemble de points qui captent la moyenne $\bar{X}_{k|k}$ et $P_{k|k}$ de l'approximation. Ces points, appelés *sigma points*, sont choisis de façon déterministe, et sont propagés à l'aide des modèles non linéaires d'état et de mesure. Le choix de ces points à l'initialisation est détaillé dans [JU97]. L'algorithme, qui fonctionne toujours récursivement avec une étape de prédiction et une de correction, est détaillé sur la figure 3.4.

L'intérêt de ce filtre par rapport au filtre de Kalman étendu est qu'il donne une estimation précise au second ordre (alors que le filtre de Kalman étendu en donnait une au premier ordre seulement), et ce quelques soient les non linéarités du système. Dans le cas où les bruits sont additifs et gaussiens, l'estimation est précise jusqu'au troisième ordre. C'est donc un filtre qui améliore les performances de l'estimation. Cependant quelques problèmes numériques subsistent, notamment durant la phase de choix des points dans laquelle des instabilités sont observées.

3.2.6 Autres méthodes

Nous avons détaillé les extensions du filtre de Kalman les plus utilisées. D'autres types de filtres ou de méthodes ont été proposées. Nous allons en citer quelques-unes brièvement. Les algorithmes de Kalman reposent sur le fait que la loi de filtrage peut être approchée par une loi normale. En pratique, lorsque la loi est multi-modale (ce qui arrive souvent), ces algorithmes sont mis en défaut. Pour remédier au problème, des filtres par mélange de lois gaussiennes ont été proposés [SA71].

Faire une approximation gaussienne est, dans de nombreux cas, une mauvaise hypothèse. Une alternative consiste alors à décrire la loi que l'on cherche par un nuage de points pondérés. Les méthodes par maillage de l'espace d'état reposent sur ce principe. Ces méthodes, qui utilisent un modèlé appelé HMM ou Hidden Markov Model, fonctionnent de la manière suivante : on effectue un maillage déterministe de l'espace d'état pour évaluer la loi de filtrage en ces points. Ces méthodes numériques sont exactes lorsque l'espace d'état est discret, constitué d'un nombre fini et faible d'états, et que les points de la grille correspondent à ces états. Les inconvénients de ce type de méthode sont l'augmentation exponentielle du coût de calcul avec la taille de l'espace

– Initialisation : $\hat{\mathbf{X}}_{0|0} = \bar{X}_0 = \mathbb{E}[\mathbf{X}_0] \quad \text{et} \quad P_{0|0} = Q_0^X = cov(\mathbf{X}_0)$

– Pour tout $k \in \mathbb{N}, \quad k \geq 1$

• A partir de l'estimation précédente $\hat{\mathbf{X}}_{k|k-1}$, $P_{k|k-1}$, on calcule le nouveau nuage pondéré de points $\{\mathbf{X}_{k-1|k-1}^{(i)}, \pi_{k-1|k-1}^{(i)}\}$

• Prédiction :

$$\hat{\mathbf{X}}_{k|k-1}^{(i)} = f_k(\hat{\mathbf{X}}_{k-1|k-1}^{(i)}) \quad \text{et} \quad \hat{\mathbf{X}}_{k|k-1} = \sum_{i=1}^{N} \pi_{k-1|k-1}^{(i)} \mathbf{X}_{k-1|k-1}^{(i)}$$

$$P_{k|k-1} = \sum_{i=1}^{N} \pi_{k-1|k-1}^{(i)} \left[(\mathbf{X}_{k|k-1}^{(i)} - \hat{\mathbf{X}}_{k|k-1})(\mathbf{X}_{k|k-1}^{(i)} - \hat{\mathbf{X}}_{k|k-1})^T \right] + Q_k^W$$

$$\hat{\mathbf{Y}}_{k|k-1}^{(i)} = h_k(\hat{\mathbf{X}}_{k|k-1}^{(i)}) \quad \text{et} \quad \hat{\mathbf{Y}}_{k|k-1} = \sum_{i=1}^{N} \pi_{k-1|k-1}^{(i)} \mathbf{Y}_{k|k-1}$$

• Correction :

$$\hat{\mathbf{X}}_{k|k} = \hat{\mathbf{X}}_{k|k-1} + K_k \left[\mathbf{Y}_k - \hat{\mathbf{Y}}_{k|k-1} \right]$$

$$P_{k|k} = P_{k|k-1} - K_k Q_k^{YY} K_k^{-1}$$

$$Q_k^{YY} = \sum_{i=1}^{N} \pi_{k-1|k-1}^{(i)} \left[(\mathbf{Y}_{k|k-1}^{(i)} - \hat{\mathbf{Y}}_{k|k-1})(\mathbf{Y}_{k|k-1}^{(i)} - \hat{\mathbf{Y}}_{k|k-1})^T \right] + Q_k^V$$

$$Q_k^{xY} = \sum_{i=1}^{N} \pi_{k-1|k-1}^{(i)} \left[(\mathbf{X}_{k|k-1}^{(i)} - \hat{\mathbf{X}}_{k|k-1})(\mathbf{Y}_{k|k-1}^{(i)} - \hat{\mathbf{Y}}_{k|k-1})^T \right] + Q_k^V$$

$$K_k = Q_k^{XY} (Q_k^{YY})^{-1}$$

FIG. 3.4 – Le filtre de Kalman unscented.

d'état, le choix de la grille et la difficulté de mise en œuvre.

En conclusion : le filtre de Kalman et ses extensions sont performants si les hypothèses sur les densités mises en jeu sont vérifées en pratique. Dans des cas plus généraux, les filtres par mélange de lois gaussiennes ou les méthodes numériques par maillage d'état sont applicables, mais sont difficilement utilisables à cause de leur coût de calcul élevé et de la difficulté à les mettre en œuvre. Plus récemment, de nouvelles méthodes ont été développées pour résoudre le problème du filtrage dans un cas général. Nous allons présenter ces méthodes dans la section suivante.

3.3 Les méthodes aléatoires : les méthodes de Monte Carlo

3.3.1 Introduction

Les filtres de Kalman présentés dans les sections précédentes assument une distribution gaussienne pour effectuer les prédictions de l'état $\mathbf{X}_k$ du procédé. Lorsque le modèle qui décrit le

système dynamique n'est plus linéaire et/ou gaussien, les méthodes non aléatoires présentées dans la section précédente ne peuvent pas, dans des temps de calculs raisonnables, fournir une estimée précise de la loi de filtrage. L'utilisation de simulations par Monte Carlo pour la construction de cette loi s'est montrée efficace pour résoudre ce problème.

Les méthodes de Monte Carlo se sont tout d'abord développées dans un cadre physique dans les années 50. Puis dans les années 90, elles ont été utilisées avec succès dans la communauté du traitement du signal. Cet essor, dû en grande partie à l'augmentation des capacités de calcul, a donné lieu à la construction d'algorithmes récursifs pour résoudre le problème du filtrage non linéaire. Dans la littérature, on trouve pour ces algorithmes les noms de filtre boot-strap (en anglais *bootstrap filter*), filtre de Monte Carlo, algorithme de condensation, ou filtre particulaire.

Ces méthodes ont été utilisées pour des applications très diverses : en traitement du signal (déconvolution de signaux, traitement de la parole), en vision par ordinateur (suivi de personnes), dans le domaine médical (diagnostique), dans le domaine financier (modélisation de données financières), mais aussi en physique, chimie et dans le domaine de la biotechnologie.

La méthode générale de ces filtres pour résoudre le problème de filtrage repose sur une approximation de la densité *a posteriori* par un ensemble de points de l'espace d'état que l'on appelle *particules*. Un poids est associé à chaque particule de manière à ce que les particules assorties de leurs poids permettent de représenter la loi conditionnelle recherchée. On fait évoluer au cours du temps ce nuage de particules ainsi que leurs poids, et on procède à une étape de sélection par rééchantillonnage. Remarquons que contrairement à la méthode du maillage décrite dans la section précédente, les particules consituent dans ce type d'approche un maillage aléatoire évolutif de l'espace d'état (et non plus un maillage déterministe).

Le filtre de Monte Carlo pondéré introduit en 1970 [Han70] fait évoluer les particules indépendamment les unes des autres en utilisant la loi d'évolution du processus d'état. La pondération des particules se fait par un calcul de vraisemblance de la mesure courante. Dans cet algorithme, il n'y a pas d'étape de sélection par rééchantillonnage ce qui implique l'utilisation d'un nombre élevé de particules et qui conduit en pratique à une dégénérescence des poids des particules (voir le paragraphe suivant pour des explications supplémentaires). Par la suite une étape de sélection par rééchantillonnage a été rajoutée à l'algorithme et, selon les auteurs, le filtre a pris le nom de filtre particulaire avec interaction pour Del Moral [dMRS92], le filtre boot-strap pour Gordon [GSS93], le filtre de Monte Carlo pour Kitagawa [Kit96] ou encore l'algorithme de condensation pour Isard et Blake [IB96a]. Dans toutes ces méthodes, les particules sont propagées selon la loi d'évolution de l'état, et rééchantillonnées à chaque étape selon un schéma classique (qui sera détaillé dans les sections suivantes).

Ces filtres présentent deux gros inconvénients. Le premier concerne la dégénérescence des poids : dans ce cas, seules quelques particules ont des poids importants. Le second concerne le fait qu'il arrive que les particules se concentrent et deviennent quasiment toutes égales. Des améliorations ont été proposées ; on pourra se référer par exemple aux travaux de Liu [Liu98], Pitt [PS99], Doucet [DGA00] ou Gilks [GB01]. Dans ces travaux, de nouvelles méthodes de rééchantillonnage sont proposées pour limiter la dégénérescence des poids.

Cette section a pour objectif de présenter ces techniques.

3.3.2 L'étape d'échantillonnage

Le principe de Monte Carlo consiste à utiliser un ensemble d'échantillons discrets obtenus par simulation pour approcher une distribution de probabilité $p(\mathbf{X})$ dont l'expression analytique est inconnue.

Lorsque l'on ne sait pas échantillonner selon la loi $p(\mathbf{X})$ mais que l'on peut évaluer cette loi en tout point à une constante près, un ensemble de réalisations peut être obtenu par l'algorithme d' *acceptation/rejet*. Pour cela, on se base sur l'hypothèse qu'il existe une constante $c < \infty$ telle que $p(\mathbf{X}) \leq c\pi(\mathbf{X})$. Cette méthode consiste à générer un échantillon $\mathbf{X}^{(i)}$ de la loi $\pi(\mathbf{X})$ et de l'accepter avec la probabilité $\frac{p(\mathbf{X}^{(i)})}{c\pi(\mathbf{X}^{(i)})}$. L'algorithme est représenté sur la figure 3.5. La fonction $\pi(\mathbf{X})$ est appelée *fonction d'importance* ou *loi de proposition* selon les articles rencontrés.

On fixe $i = 0$.
Tant que $i \neq N$

1. On génère une réalisation $\mathbf{X}^{(i)}$ de la loi $\pi(\mathbf{X}^{(i)})$
2. On génère une réalisation u de la loi uniforme $\mathcal{U}_{(0,1)}$
3. Si $u < \frac{p(\mathbf{X}^{(i)})}{c\pi(\mathbf{X}^{(i)})}$ alors on accepte $\mathbf{X}^{(i)}$ et on incrémente i. Sinon, on rejette l'échantillon.

FIG. 3.5 – L'algorithme acceptation/rejet.

Une autre méthode, l'*échantillonnage pondéré* est une alternative à l'algorithme d'acceptation/rejet. Elle est utilisée lorsque l'hypothèse sur l'existence de c est relachée. On suppose pouvoir tirer des échantillons selon une fonction d'importance $\pi(\mathbf{X})$ dont le support inclut le support de $p(\mathbf{X})$. Pour chacune des réalisations $\mathbf{X}^{(i)}$ de la loi $\pi(\mathbf{X})$, on évalue les poids d'importance associés $\tilde{w}^i = \frac{w^i}{\sum_{j=1}^{N} w^j}$. L'algorithme est donné sur la figure 3.6.

1. On génère N réalisations $\mathbf{X}^{(i)}$ de la loi $\pi(\mathbf{X})$
2. On affecte à chaque réalisation un poids $\tilde{w}^i = \frac{w^i}{\sum_{j=1}^{N} w^j}$
   où $w^i = \frac{p(\mathbf{X}^{(i)})}{\pi(\mathbf{X}^{(i)})}$

FIG. 3.6 – L'algorithme d'échantillonnage pondéré.

L'échantillonnage pondéré est la première étape dans les méthodes séquentielles de Monte Carlo. En théorie, cet algorithme est valide quelque soit la loi de proposition utilisée. Cependant, il est d'autant plus efficace que la fonction d'importance $\pi(\mathbf{X})$ est proche de $p(\mathbf{X})$. Plusieurs stratégies sont possibles ; nous allons brièvement les présenter.

La solution la plus simple consiste à propager les particules selon la loi d'évolution *a priori*. Elle correspond aux premiers filtres particulaires développés. En particulier le filtre boot-strap proposé par Gordon [GSS93] associe cette étape d'échantillonnage pondéré avec un rééchantillon-

nage systématique (voir le paragraphe spécifique pour le rééchantillonnage).

D'autres ont proposé de rajouter une deuxième étape de Monte Carlo, ou combinent filtrage particulaire et itérations de Monte Carlo, comme Neal [Nea01], Musso [MOG01] ou Godsil [GC01].

Par analogies aux extensions du filtre de Kalman, certains ont proposé des versions du filtre particulaire, appelées *filtre particulaire étendu* et *filtre particulaire unscented*. L'intérêt de ces méthodes est de prendre en compte la vraisemblance lors de la propagation des particules. On peut se référer aux travaux de Doucet [DGA00] et Merwe [MDFW00] pour plus de détails.

On peut également citer les travaux de Pitt [PS99, PS01] concernant le *filtre particulaire auxiliaire* dans le but de réduire le problème de dégénérescence lorsque la vraisemblance est très informative.

3.3.3 Des problèmes et des solutions

Le problème de dégénérescence des poids des particules a été évoqué dans l'introduction de la section. Il est apparu après la mise en place des méthodes séquentielles de Monte Carlo, qui ne rééchantillonnaient pas le nuage de particules à chaque pas de temps. Ce problème conduit en pratique à une divergence du nuage de particules, et donc à une mauvaise estimation en sortie de la loi recherchée.

Une des solutions proposées pour résoudre le problème de la dégénérescence des poids est d'ajouter une étape de rééchantillonnage des particules à chaque pas de temps. Le principe de cette étape est de supprimer les particules qui ont un faible poids normalisé et de dupliquer celles qui ont un poids fort. La technique classique est de tirer avec remise N nouvelles particules parmi l'ensemble des particules proportionnellement à leur poids, mais d'autres méthodes existent et seront décrites dans les paragraphes suivants. L'algorithme, muni de cette nouvelle étape, constitue l'algorithme du *filtrage particulaire*.

L'ajout de cette étape de rééchantillonnage permet d'améliorer la qualité des estimations tout en limitant le problème de la dégénérescence des poids. Cependant, plusieurs problèmes apparaissent : l'algorithme initial était parallélisable très facilement, alors que le nouveau ne l'est plus. D'autre part, le fait de dupliquer certaines particules introduit (de manière théorique) une dépendance entre les particules, ce qui pose problème lorsque l'on tente d'établir des résultats de convergence de l'algorithme. Enfin, le dernier problème, évoqué également en introduction) concerne l'appauvrissement des états : les particules de poids très élevés sont sélectionnées plusieurs fois, et cela peut, dans certains cas, aboutir à la perte de modes de la densité. Si on considère un cas extrême, toutes les particules se regroupent à la même position et se valent.

3.3.4 L'étape de rééchantillonnage

Comme on l'a expliqué précédemment, un algorithme n'utilisant qu'une étape d'échantillonnage ne permet pas, dans un grand nombre de cas, d'avoir une estimation valide de la loi *a posteriori* à cause du problème de la dégénérescence des poids. L'ajout d'une étape de rééchantillonnage des particules permet de limiter cette dégénérescence, en supprimant les particules de

poids faibles et en dupliquant celles qui ont des poids forts. De nombreuses stratégies ont été proposées ; nous allons en présenter quelques-unes.

Rééchantillonnage multinomial

L'idée générale de cette méthode est de tirer avec remise N nouvelles particules $\tilde{\mathbf{X}}_k^{(i)}$ parmi l'ensemble $\left\{\mathbf{X}_k^{(i)}\right\}_{i=1...N}$ proportionnellement aux poids $\left\{\tilde{w}_k^{(i)}\right\}_{i=1...N}$. Les nouvelles particules correspondent à un ensemble d'échantillons indépendants et identiquement distribués de l'approximation discrète de la loi *a posteriori* [LC95, GSS93]. Leurs poids sont remis à $1/N$.

1. On calcule la suite des poids d'importance normalisés cumulés
$$Wc_i = \sum_{j=1}^{i} = \tilde{w}_k^j$$
2. Pour i = 1 ... N
   (a) On génère $u_i \sim \mathcal{U}_{[0,1]}$
   (b) On calcule l'indice j tel que $Wc_{j-1} < u_i \leq Wc_j$
   (c) On pose $\tilde{\mathbf{X}}_k^j = \mathbf{X}_k^j$
3. Pour i = 1 ... N, on pose $\mathbf{X}_k^i = \tilde{\mathbf{X}}_k^i$ et $\tilde{w}_k^{(i)} = 1/N$

FIG. 3.7 – L'algorithme de rééchantillonnage multinomial.

Encore appelée méthode de rééchantillonnage multinomial classique, cette méthode de rééchantillonnage est la plus utilisée à cause de sa simplicité de mise en œvre. La variance de l'algorithme est $var(N_i) = N\tilde{w}_k^{(i)}(1 - \tilde{w}_k^{(i)})$. Enfin, cette procédure peut être implémentée selon une complexité $\mathcal{O}(N)$.

Rééchantillonnage résiduel

Contrairement à la méthode précédente, cette méthode est partiellement déterministe. Elle se décompose en deux étapes. Tout d'abord, pour chacune des particules $\mathbf{X}_k^{(i)}$, on attribue un nombre de descendants $\tilde{N}_i$ proportionnel au poids de la particule $\tilde{w}_k^{(i)}$, que l'on prend égal à la partie entière de $N\tilde{w}_k^{(i)}$. Puis, pour conserver un nombre de particules constant, on utilise une procédure de rééchantillonnage systématique pour tirer les $N' = N - \sum_{i=1}^{N} \tilde{N}_i$ particules restantes proportionnellement aux nouveaux poids $\tilde{w}_k^{'(i)} = (N\tilde{w}_k^{(i)} - \tilde{N}_i)/N'$. La variance de cette méthode est $var(N_i) = N'\tilde{w}_k^{'(i)}(1 - \tilde{w}_k^{'(i)})$, ce qui est inférieur à la variance de la procédure classique.

Rééchantillonnage à variance minimale

La méthode de rééchantillonnage à variance minimale (ou systématique) procède de la même manière que le rééchantillonnage multinomial. Cependant, au lieu de traiter une suite de u_i tirés indépendamment selon une loi $\mathcal{U}_{[0,1]}$, on utilise une suite u_i définie par :

$$u_1 \sim \mathcal{U}_{[0,1/N]} \qquad u_i = u_1 + i/N$$

L'intérêt de cette méthode est le gain en temps de calcul, puisqu'une seule variable doit être tirée. D'autre part, le nombre de descendants pour chaque particule ne diffère pas de $N\tilde{w}_k^{(i)}$ à

± 1 près. La variance de la procédure est minimale.

Il existe d'autres méthodes à variance minimale comme les méthodes de rééchantillonnage par mécanisme de branchement (appelé en anglais *branching correction*). Mais dans ce type de procédure, le nombre de particules varie. Nous ne détaillerons pas plus ce type de méthode. On peut se référer aux travaux de Crisan pour plus de détails [CG99].

La question qui se pose alors est celle de la fréquence d'utilisation de la procédure. Faut-il effectuer un rééchantillonnage à chaque pas de temps ou seulement de temps en temps ? Comment savoir alors si le moment est bien choisi pour rééchantillonner ? Une solution possible consiste à utiliser la *taille efficace du N-échantillon*, notée ESS (Effective Sample Size) définie par :

$$ESS = \frac{N}{\mathbb{E}_{\pi(.|\mathbf{Y}_{1:k})}\left[w_k^2\right]}$$

Kong [KLW94] et Liu [LC95] montrent que la taille efficace du N-échantillon permet de mesurer la dégénérescence de l'algorithme. L'expression théorique ci-dessus ne peut pas être calculée directement ; en pratique, on l'approche par l'estimateur suivant :

$$ESS_N = \frac{N}{\sum_{i=1}^{N}(\tilde{w}_k^{(i)})^2}$$

où les $\tilde{w}_k^{(i)}$ sont les poids d'importance normalisés. Une valeur faible de ESS_N correspond à une forte dégénérescence de l'échantillon. Ainsi, à chaque itération on calcule la taille efficace du N-échantillon et on prend la décision de rééchantillonner ou pas.

D'autres méthodes ont été proposées. On pourra se référer aux travaux de thèse de Hue [Hue03] pour des détails sur ces méthodes. D'autre part, on y trouvera une série d'expériences montrant qu'un rééchantillonnage adaptatif est plus performant qu'un rééchantillonnage systématique dans le cas du pistage mono cible par mesure d'angle seul.

3.4 Conclusions

Nous avons présenté dans ce chapitre les méthodes de Monte Carlo. Ces méthodes séquentielles mises en place pour l'analyse de systèmes non linéaires et/ou non gaussiens ont fait l'objet de nombreuses recherches depuis les années 1990. Ces méthodes permettent de répondre au problème du filtrage non linéaire avec des performances supérieures à celles des algorithmes non aléatoires dans le cas général. Ces méthodes reposent sur une approximation de la loi *a posteriori* en un nuage pondéré de particules. Ces particules sont échantillonnées et mises à jour par tirages aléatoires, selon des équations induites par le modèle considéré.

Dans ce chapitre nous avons présenté les méthodes de Monte Carlo dans un cadre théorique. L'échantillonnage permet de construire de manière empirique une loi $p(X)$ à partir de tirages selon une loi d'importance $\pi(X)$. L'estimation est d'autant meilleure que la loi d'importance est proche de la loi $p(X)$. Dans le cadre bayésien, la loi $p(X)$ est $p(X \mid Y)$. Pour obtenir une solution séquentielle, on suppose que la loi d'importance (qui permet de faire évoluer les particules dans l'espace d'état) a une forme récursive. En pratique, les erreurs dues au fait que la loi d'importance $\pi(X)$ n'est pas précisément la loi $p(X)$ mènent à une dégénérescence des particules. L'introduction d'une étape de rééchantillonnage permet de limiter cette dégénérescence,

en dupliquant les particules de poids forts et en supprimant celles de poids faible. Mais effectué systématiquement, ce rééchantillonnage peut induire une mauvaise exploration de l'espace d'état.

Le filtrage particulaire est une méthode de Monte Carlo particulière. Une itération de cet algorithme correspond à une étape d'échantillonnage pondéré (que nous appellerons par la suite étape de propagation) suivie (dans la plupart des cas) d'une étape de rééchantillonnage. Nous allons, dans le chapitre suivant, présenter trois algorithmes de filtrage particulaire dans le cadre du suivi d'objet dans une séquence d'image : le premier utilise des histogrammes en niveaux de gris, le second est basé sur des modèles géométriques (2D et 3D) et le dernier intègre à la fois les informations de ninveaux de gris et celles des contours des images pour rendre l'algorithme plus robuste.

Chapitre 4

Algorithmes de suivi proposés

Sommaire

4.1 Introduction

Dans les chapitres précédents, nous avons décrit les différentes méthodes utilisées pour effectuer le suivi d'objets dans une séquence vidéo, et présenté le problème du filtrage. Le chapitre 2 a mis en évidence les difficultés auxquelles il faut faire face pour que le suivi soit robuste. Les principales difficultés résident dans les changements d'illumination de la scène, les changements d'apparence de l'objet d'intérêt ainsi que les problèmes d'occultation. Le chapitre 3 a mis en évidence les différentes approches disponibles pour résoudre le problème du filtrage, qui varient selon les hypothèses de linéarité (ou non linéarité) des modèles et les hypothèses de loi gaussiennes ou non.

L'objectif de ce chapitre est de présenter trois algorithmes dont l'application est le suivi d'objets dans une scène. Nous nous plaçons dans un cadre bayésien, et nous formulons le problème du suivi de manière probabiliste, au moyen d'un filtre particulaire. Le premier algorithme que nous développerons est celui d'un suivi basé sur des histogrammes de niveau de gris ; c'est un suivi purement 2D dans le sens où aucune information 3D réelle de la scène n'est utilisée. Le second algorithme quant à lui permet d'intégrer de l'information 3D réelle de la scène observée.

Le suivi est basé sur des modèles géométriques (qui peuvent être 2D ou 3D), que nous combinons au filtrage particulaire. Nous ajoutons une étape supplémentaire au filtrage particulaire classique, nous permettant de changer de modèle lorsque l'algorithme le juge nécéssaire. Cela nous permet, dans le cadre d'une application militaire dans laquelle la caméra est embarquée sur un avion, de suivre un objet dans une scène où les changements d'apparence, d'échelle, et d'illumination peuvent être très grands. Enfin, le dernier algorithme intègre à la fois les informations de niveaux de gris et celles de contours dans un filtrage particulaire. L'algorithme obtenu est plus robuste, en particulier lorsque les contours de l'objet d'intérêt ne sont pas extraits correctement ou qu'au voisinage de l'objet d'intérêt la densité de contours extraits est grande. Les résultats des tests effectués sur les différents algorithmes sont présentés dans les deux chapitres suivants.

4.2 Le filtrage particulaire

Comme nous l'avons vu dans le chapitre précédent, la méthode du filtrage particulaire consiste à représenter la loi conditionnelle de l'état de l'objet par un ensemble d'échantillons associés à des poids. Chaque échantillon, aussi appelé *particule*, est un élément qui représente un état hypothétique de l'objet $\mathbf{X}$, auquel on associe un poids w, représentant sa vraisemblance par rapport aux données (les images dans notre cas), et éventuellement à un modèle de référence. On peut écrire l'ensemble de la façon suivante :

$$S = \left\{ \left(\mathbf{X}^{(i)}, w^{(i)} \right), i = 1, ..., N \right\} \text{ où } \sum_{i=1}^{N} w^{(i)} = 1$$

L'évolution de l'ensemble est obtenu en propageant chacun des échantillons selon un modèle de mouvement. On attribue ensuite un poids à chaque particule en fonction des observations, et on détermine l'état moyen du système à chaque étape par :

$$E[S] = \sum_{i=1}^{N} w^{(i)} \mathbf{X}^{(i)} \tag{4.1}$$

Si l'on formule ce qui a été dit précédemment, en notant $\mathbf{X}_k$ le vecteur d'état de l'objet à l'instant k, et $\mathbf{Y}_{0:k}$ les observations du temps $k = 0$ jusqu'au temps k, il s'agit de déterminer l'état du système à l'instant k en fonction des observations et de l'état à l'instant précédent ; on cherche donc $p(\mathbf{X}_k \mid \mathbf{Y}_k, \mathbf{X}_{k-1})$.

La sortie du filtre à l'instant k est une approximation d'une distribution de probabilité de vecteurs d'état d'objets, contenant les informtations de position, taille et orientations de l'objet ; cette distribution est donnée par l'ensemble des particules. C'est un processus itératif qui fonctionne de la manière suivante : la sortie à l'instant $k-1$ est l'ensemble des particules pondérées $S_{k-1} = \{(\mathbf{X}_{k-1}^{(i)}, w_{k-1}^{(i)}), i = 1, ..., N\}$, où $\mathbf{X}_{t-1}^{(i)}$ est le vecteur qui décrit l'état de la $i^{\text{ème}}$ particule et $w_{k-1}^{(i)}$ son poids.

Chaque particule est soumise à une étape de prédiction, qui correspond à une étape de propagation selon un modèle dynamique du système (un modèle de mouvement et/ou du bruit par exemple). Puis on applique l'étape d'observation, qui a pour but de calculer le poids de chacune des particules en évaluant sa vraisemblance. On obtient donc en sortie la nouvelle représentation de l'ensemble des particules $S_k = \{(\mathbf{X}_k^{(i)}, w_k^{(i)}), i = 1, ..., N\}$ à l'instant k.

La dernière opération est le rééchantillonnage des particules, qui permet de concentrer les particules dans la région d'intérêt. Les méthodes de regénération de particules sont nombreuses, mais elles ne sont pas encore toutes bien connues et maîtrisées pour le moment. L'étape de rééchantillonnage systématique (voir le paragraphe 3.3.4) est la plus employée. On peut se référer à [IB96b] pour plus de détails sur les différentes méthodes existantes.

L'algorithme du filtrage particulaire, dans sa version la plus simple, est résumé sur la figure 4.1.

Etant donné l'ensemble des particules $\{(\mathbf{X}_{k-1}^{(i)}, w_{k-1}^{(i)}), i = 1, ..., N\}$ à l'instant $k-1$, l'objectif est de construire le nouvel ensemble $\{(\mathbf{X}_k^{(i)}, w_k^{(i)}), i = 1, ..., N\}$ à l'instant k.

Pour chaque particule i de l'ensemble :

1. On propage la particule selon le modèle dynamique du système pour obtenir $\mathbf{X}_k^{(i)}$
$$p(\mathbf{X}_k \mid \mathbf{X}_{k-1} = \mathbf{X}_{k-1}^{(i)})$$
2. On mesure et pondère le nouvel état de la particule selon les observations effectuées $\mathbf{Y}_k$
$$w_k^{(i)} = p(\mathbf{Y}_k \mid \mathbf{X}_k = \mathbf{X}_k^{(i)})$$
et on normalise les poids pour que
$$\sum_{i=1}^{N} w_k^{(i)} = 1$$
3. On fait l'étape de rééchantillonnage pour assurer la concentration des particules dans les régions d'intérêt.

FIG. 4.1 – Le filtrage particulaire.

4.3 Un algorithme de suivi par histogrammes

4.3.1 La problématique

Une des principales difficultés du suivi d'objets dans une séquence vidéo réside dans la prise en compte des changements d'apparence (non seulement ceux de la cible, mais aussi ceux qui proviennent de la scène directement, les changements d'illumination par exemple). Le cadre bayésien est très flexible et ses performances dans le domaine du suivi d'un ou plusieurs objets ont été prouvées. D'autre part, les histogrammes de niveaux de gris ou de couleurs permettent de suivre un objet avec une complexité calculatoire faible. Les algorithmes de suivi d'objets reposant sur des histogrammes de couleurs intégrés dans un cadre bayésien [JFEM01, NKMG03] se sont montrés performants pour une application donnée (le suivi de visage par exemple), mais ne peuvent pas être généralisés facilement, les paramètres et l'initialisation de l'algorithme étant

spécifiques à la séquence d'entrée.

Notre premier algorithme est basé sur des histogrammes de niveaux de gris combinés à un filtrage particulaire qui permet de résoudre un certain nombre des problèmes traditionnels du suivi d'objets (occultations, changements d'apparence de l'objet, changements d'échelle ou d'illumination de la scène), et de s'adapter facilement à la séquence d'entrée (suivi de joueur de football, de structures dans des images aériennes, de visages). La nouveauté de l'algorithme réside dans sa capacité à fixer automatiquement tous les paramètres nécessaires au suivi. Notre contribution est donc d'automatiser un algorithme de telle façon à le rendre performant non plus pour une application unique, mais pour des applications diverses, sans avoir à faire manuellement la sélection des paramètres de manière à rendre l'algorithme performant pour l'application désirée.

4.3.2 Combiner les histogrammes de couleur et filtrage particulaire

Ce paragraphe a pour objectif d'expliquer comment on combine les histogrammes (voir le paragraphe 2.3.6) de couleur (ou de niveaux de gris) à un filtrage particulaire, et de présenter l'algorithme que nous avons développé.

Le choix du modèle : les histogrammes de couleur

Comme il a été dit précédemment, nous avons choisi d'utiliser les distributions de couleurs pour modéliser l'objet pour de nombreuses raisons : invariance aux rotations, robustesse aux occultations partielles, et à l'aspect rigide/non rigide de l'objet à suivre. On suppose que les distributions sont discrétisées en K classes (voir ci-dessous pour la sélection automatique de K). Dans notre approche, l'objet est modélisé par une ellipse (de nombreuses approches utilisent des rectangles mais il nous a paru plus intéressant d'utiliser une ellipse pouvant tourner) ; on peut calculer les histogrammes dans l'espace RVB (ou dans n'importe quel autre espace) des couleurs, ou plus simplement en niveaux de gris, en fonction de la séquence d'entrée.

Pour limiter la perte d'information spatiale due aux histogrammes, Nummiaro [NKMG03] et Pérez [PHVG02] ont attribué des poids différents aux pixels dans l'ellipse. Leur raisonnement est le suivant : plus les pixels sont loin du centre de l'ellipse, plus on leur associe un poids faible. La fonction de poids correspondante est donnée par l'expression suivante :

$$f(d) = \begin{cases} 1 - d^2 & \text{si } d \leq 1 \\ 0 & \text{sinon} \end{cases}$$

où d est la distance normalisée (c'est-à-dire divisée par le demi grand axe de l'ellipse) du pixel au centre de l'ellipse. Notons que d'autres fonctions de poids peuvent être utilisées : Comaniciu, par exemple dans [CRM03], utilise le noyau d'Epanechnikov.

On calcule une distribution de couleur ou de niveau de gris $p_{\mathbf{x}} = \{p_{\mathbf{x}}^{(j)}\}_{j=1,\ldots,K}$ à un pixel donné $\mathbf{x}$ par :

$$p_{\mathbf{x}}^{(j)} = C \sum_{\mathbf{x}_i \in E} k\left(\frac{\|\mathbf{x} - \mathbf{x}_i\|}{\sqrt{(l_x^2 + l_y^2)}}\right) \delta\left(h(\mathbf{x}_i) - j\right) \tag{4.2}$$

où δ est la fonction delta de Kronecker, E est l'ensemble des pixels contenus dans l'ellipse, l_x et l_y les demi axes de l'ellipse, $h(\mathbf{x}_i)$ attribue une des K classes de l'histogramme d'une couleur

donnée au pixel $\mathbf{x}_i$ et C est le facteur de normalisation, qui assure que $\sum_{j=1}^{K} p_{\mathbf{x}}^{(j)} = 1$.

L'expression de C est donnée par :

$$C = \left[\sum_{\mathbf{x}_i \in E} k \left(\frac{\|\mathbf{x} - \mathbf{x}_i\|}{\sqrt{(l_x^2 + l_y^2)}} \right) \right]^{-1}$$

La ressemblance entre deux distributions p et q est donnée par le coefficient de Bhattacharyya :

$$\rho[p,q] = \sum_{j=1}^{K} \sqrt{p^{(j)} q^{(j)}} \tag{4.3}$$

Pour deux distributions identiques, on a $\rho = 1$, ce qui correspond à une correspondance parfaite. On utilise la distance de Bhattacharyya $d_b = \sqrt{1 - \rho[p,q]}$ dans notre algorithme pour évaluer la distance entre deux histogrammes. Notons que d'autres mesures de distance entre histogrammes existent [CS02], et que la distance de Bhattacharyya présente le meilleur compromis entre complexité calculatoire et performances.

L'algorithme

On veut suivre une région d'intérêt dans le plan image. Nous avons choisi de paramétrer cette région par une ellipse, définie par :

$$\mathbf{X} = \left\{ x, y, \dot{x}, \dot{y}, \theta, l_x, l_y, \dot{l}_x, \dot{l}_y \right\}$$

où x et y représentent les coordonnées du centre de l'ellipse, $\dot{x}$ et $\dot{y}$ les vitesses en x et y du centre de l'ellipse, θ l'orientation de l'ellipse, l_x et l_y ses demi axes, et $\dot{l}_x$ et $\dot{l}_y$ les vitesses de l_x et l_y. Notons que les paramètres de l'ellipse peuvent varier indépendamment les uns des autres ce qui rend ce modèle flexible.

Pour propager une particule, nous utilisons un modèle de mouvement au premier ordre :

$$\mathbf{X}_t = \mathsf{A}\mathbf{X}_{\mathbf{t-1}} + \mathbf{b}_{\mathbf{t-1}} \tag{4.4}$$

où $\mathbf{X}_t$ est une variable aléatoire gaussienne et A une matrice dont le but est de décrire un objet se déplacant à vitesse constante en x, y, l_x et l_x. A s'exprime donc de la manière suivante :

$$\mathsf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

L'algorithme fonctionne de la manière suivante : nous initialisons à la main l'ellipse sur la première image, calculons la distribution du modèle, et initialisons l'ensemble des particules.

Puis, pour chaque image de la séquence, les particules sont propagées en utilisant le modèle de mouvement défini précédemment (équation 4.4). Pour chaque particule, nous calculons la distance de Bhattacharyya entre la distribution du modèle et celle de l'état hypothétique correspondant à cette particule. Cette distance est utilisée pour calculer le poids $w^{(i)}$ associé à la particule i. Nous pouvons remarquer que les poids associés à chacune des particules favorisent les particules dont les distributions sont ressemblantes à celle du modèle. Les poids sont calculés en utilisant l'équation :

$$w^{(i)} = \exp(-\beta d_b^{(i)}) \tag{4.5}$$

pour chaque particule i de l'ensemble, où β est une constante fixée, et $d_b^{(i)}$ représente la distance de Bhattacharyya entre la $i^{\text{ème}}$ particule et le modèle de l'objet.

La dernière étape consiste à rééchantillonner les particules pour assurer la bonne évolution de l'ensemble des particules, et de déterminer l'état moyen de l'objet . Lors de cette étape de rééchantillonnage, les particules sont éliminées ou dupliquées selon leur poids : plus leur poids est élevé, plus on duplique la particule. Dans la littérature, différentes méthodes existent pour rééchantillonner un ensemble de particules, et on peut se référer à [Leg03] pour plus de détails ; nous avons choisi d'utiliser un rééchantillonnage systématique pour sa simplicité. (On peut se référer au chapitre précédent pour les détails d'implémentation de cette méthode de rééchantillonnage).

Efin, en ce qui concerne la complexité de notre algorithme, elle est linéaire par rapport au nombre de particules utilisées, et linéaire aussi par rapport à la taille de l'ellipse. Les opérations sont simples et ne nécessitent pas de gros calculs, ce qui permet de rendre l'algorithme implémentable pour faire du temps réel : la propagation des particules est une multiplication de matrice, le remplissage des histogrammes correspond au nombre de pixels de l'ellipse, la comparaison d'histogrammes est linéaire par rapport au nombre de classes des histogrammes, et l'opération de rééchantillonnage est en $O(nbparticules)$.

4.3.3 Nos contributions

De nombreuses approches reposant sur des histogrammes de couleurs intégrés à un cadre probabiliste ont été proposées pour le suivi d'objets (soumis ou non à des déformations). Mais aucune de ces approches n'est assez flexible pour régler les paramètres du suivi automatiquement de manière à rendre l'algorithme utilisable facilement pour diverses applications. Nous proposons dans cette partie des critères permettant de régler les paramètres du suivi automatiquement.

La sélection automatique du nombre de classes de l'histogramme

Le nombre de classes nécessaires pour calculer les histogrammes est un paramètre important et doit être déterminé automatiquement. La discrétisation en un nombre de classes trop important ne permet pas d'être robuste aux changements d'illumination de la scène, aux changements d'apparence de l'objet ou au bruit, et la plupart du temps, l'algorithme dérive au fur et à mesure de la séquence. A l'inverse, un nombre de classes trop faible ne permet pas de discriminer efficacement l'objet du reste de la scène, et le suivi échoue. Cette intuition/remarque a été confirmée par nos expériences, comme le montrent les résultats de notre évaluation présentés ci-après.

Dans la plupart des approches existantes, le nombre de classes paraît être choisi arbitrairement au début de la séquence et reste fixé tout au long du suivi. Rien n'indique qu'une telle partition est optimale étant donnée la densité que l'on veut estimer. S'il était possible de trouver la partition optimale, l'algorithme serait plus robuste.

De nombreuses recherches dans le domaine de l'estimation et de la sélection de modèles ont été réalisées dans le but de résoudre le problème suivant : comment déterminer le modèle optimal à partir des données disponibles ? On considère qu'un modèle est d'autant meilleur qu'il est peu complexe et fidèle aux données. On parle donc de modèle optimal au sens de ce compromis à faire (entre complexité et fidélité aux données). Généralement les méthodes obtenues sont basées sur des considérations asymptotiques, et ne marchent pas bien dans le cas où les données sont peu nombreuses. De plus, beaucoup d'entre elles font l'hypothèse d'information *a priori* sur la densité à estimer. Récemment, Birgé et Rozenholc [BR02] ont généralisé l'estimateur d'Akaike. Le critère d'Akaike [Aka73] est une mesure statistique pour la sélection de modèle qui stipule que si deux modèles représentent aussi bien les données, le modèle le plus simple sera le meilleur prédicteur. Dans ce qui suit, nous résumons brièvement leur méthode pour déterminer le nombre optimal de classes pour nos histogrammes. Pour les détails théoriques sous-jacents, on peut se référer à [BR02].

L'objectif est de trouver un estimateur pour notre histogramme $\hat{f}$ basé sur une partition $\{I_1, ..., I_K\}$ de $[0,1]$ en K intervalles de même longueur. $X_1, X_2, ..., X_n$ sont n échantillons de la densité inconnue f que l'on cherche à estimer. K est donné par :

$$K = \arg\max_K (L_n(K) - \text{pénalité}(K)) \tag{4.6}$$

où $L_n(K)$ est le log de la vraisemblance de l'histogramme avec K classes, donné par :

$$L_n(K) = \sum_{j=1}^{K} M_j \log(\frac{KM_j}{n}) \quad \text{avec} \quad M_j = \sum_{i=1}^{n} \mathbf{1}_{I_j}(X_i)$$

où $\mathbf{1}_{I_j}$ est la fonction indicatrice définie par :

$$\mathbf{1}_{I_j}(x) = \begin{cases} 1 & \text{si } x \in I_j \\ 0 & \text{sinon} \end{cases}$$

En pratique, on détermine $\arg\max_K(L_n(K) - \text{pénalité}(K))$ en effectuant les calculs pour plusieurs valeurs de K et en ne gardant que la meilleure.

La fonction de pénalité est donnée par :

$$\text{pénalité}(K) = K - 1 + (log(K))^{2.5} \quad \text{pour } K \geq 1$$

Cette approche est donc un exemple typique de méthode de sélection de modèle, mettant en évidence le compromis à faire entre la complexité du modèle et sa fidélité aux données.

Incorporer de l'information spatiale

Le problème avec l'utilisation d'histogrammes est que l'information spatiale est perdue, à l'inverse des approches de type « template matching » qui utilisent toute l'information spatiale. Comme il a été dit précédemment, attribuer des poids différents aux pixels selon la distance qui

les sépare du centre de l'ellipse permet de compenser cette perte d'information. Pour conserver davantage la relation spatiale existante entre les pixels voisins de la région d'intérêt, nous avons décidé de diviser notre ellipse en 4 quartiers, et de procéder pour chaque quartier de la même façon que pour l'ellipse entière décrite précédemment. La division de l'ellipse accroît la robustesse du suivi puisque l'on dispose de 4 mesures de similarité entre une hypothèse et le modèle, qui sont facilement combinables ; cela permet une meilleure discrimination de l'objet par rapport au reste de la scène.

Un autre avantage de la division de l'ellipse est de pouvoir utiliser le critère pour la sélection automatique du nombre de classes à chacun des quartiers. Le nombre de classes peut être différent dans chacun des quartiers de l'ellipse selon la quantité de données disponible (un quartier de l'ellipse contenant une région homogène n'a pas besoin d'autant de classes qu'un quartier très texturé).

Enfin, la division de l'ellipse en quartiers permet de détecter et/ou gérer les occultations partielles plus facilement : si le coefficient de Bhattacharyya est mauvais pour un des quartiers mais très bon pour les autres, alors on détecte une occultation partielle. Pour évaluer la vraisemblance de la particule, on combine les 4 mesures en calculant la valeur médiane. On peut aussi imaginer d'autres partitions que les quartiers, et de même, effectuer une sélection automatique de la meilleure partition.

La mise à jour du modèle

La couleur apparente d'un objet peut varier au cours du temps ; cela peut être dû à des changements d'illumination de la scène, à des modifications des paramètres de la caméra ou au mouvement de l'objet. Pour faire face à ces changements d'apparence, on a besoin de mettre à jour le modèle. Le filtrage particulaire a déjà été utilisé avec des modèles statiques [HH98] ou adaptatifs [NKMG02]. La plupart du temps, le modèle est mis à jour à chaque fois que la probabilité de l'état moyen de l'objet calculé est au-dessus d'un seuil fixé arbitrairement à l'initialisation. Le risque de cette méthode est qu'elle peut dériver progressivement de l'objet initial.

L'idée que nous proposons est la suivante : pourquoi ne pourrait-on pas mettre à jour le modèle lorsqu'il n'est plus bon ? Le modèle devrait être mis à jour lorsque l'apparence du modèle a suffisamment changé. Ainsi, nous utilisons le critère suivant : si l'état moyen est en dessous d'un seuil π_T (voir le paragraphe suivant pour la détermination de ce seuil), alors le modèle est mis à jour en utilisant l'équation suivante :

$$p_{E[S_k]} \leq \pi_T \Rightarrow \mathbf{X}_k^j = (1-\alpha)\mathbf{X}_{k-1}^j + \alpha p_{E[S_k]}^j \tag{4.7}$$

Comment fixer le seuil π_T ? Ce seuil dépend de plusieurs paramètres plus ou moins connectés : la taille de l'ellipse et le nombre de classes de l'histogramme. En général, plus l'ellipse est grande, plus le nombre de classes est grand. Le coefficient de Bhattacharyya est alors plus faible, et le seuil pour la mise à jour du modèle doit être plus bas lui aussi. Pour régler ce seuil automatiquement, on fait l'hypothèse qu'au début de la séquence (les premières images), l'état moyen de l'objet est bien estimé, et on fixe le seuil de manière empirique : $\pi_T = \rho - c$ où c est une constante.

Notons que cette méthode de mise à jour n'est pas robuste aux occultations (au contraire de la mise à jour classique, qui se fait lorsque le modèle est suffisamment bon). Cependant, sur les séquences de tests que nous avons utilisé, il n'y avait pas d'occultations. Et dans ce cas, les

résultats du suivi utilisant notre méthode de mise à jour étaient meilleurs. C'est pourquoi nous l'avons conservée.

Le schéma global de l'algorithme est donné Figure 4.2.

Ce travail a fait l'objet d'une présentation d'un poster au workshop IEEE Motion 2005 à Breckenridge, Colorado en Janvier 2005 et a été publié [JSR05].

1. Initialisation :
   - On sélectionne l'ellipse dans la première image, que l'on partage en 4 quartiers.
   - On détermine automatiquement le nombre de classes de chaque histogramme selon l'équation (4.6)
   - On calcule la distribution du modèle dans chaque quartier de l'ellipse avec l'équation (4.2)
   - On initialise l'ensemble des particules (N copies identiques de l'ellipse)
2. Pour chaque nouvelle image :
   - On propage l'ensemble des particules en utilisant le modèle dynamique selon l'équation (4.4)
   - Pour chaque particule $\mathbf{X}_t$, on calcule
     - La distribution de couleur avec l'équation (4.2)
     - Le coefficient de Bhattacharyya avec (4.3)
     - Le poids avec l'équation (4.5)
   - On estime l'état moyen selon l'équation (4.1)
   - On rééchantillonne (voir Paragraphe 2.3)
   - On met à jour le modèle si besoin avec (4.7)

FIG. 4.2 – L'algorithme de suivi basé sur des histogrammes de couleur.

4.4 Un algorithme de suivi utilisant des modèles géométriques

4.4.1 La problématique

Dans un contexte militaire où la caméra est embarquée sur un avion, la problématique du suivi d'objets (structures ou véhicules, fixes ou mobiles) dans une séquence d'images est différente du cas où la caméra utilisée est une caméra « domestique ». La caméra en mouvement se déplace à des vitesses bien supérieures que dans le cas d'applications usuelles pour lesquelles elle est généralement tenue par un utilisateur, les changements d'échelles sont plus grands et le ratio « taille de l'objet / distance de la caméra à l'objet » pose souvent problème.

Les techniques de filtrage disponibles aujourd'hui dans la littérature permettent une grande flexibilité quant au choix des modèles ou des fonctions pour effectuer du suivi d'objets dans une séquence d'images. Dans certaines approches, plusieurs modèles de mouvement sont utilisés pour améliorer les performances du suivi d'un objet en mouvement en ajustant le modèle de mouvemet

au mouvement de l'objet [IB98, KR04]. Par contre, à notre connaissance, aucune approche de suivi n'a été développée permettant de changer de modèle au cours du suivi.

Le fait de pouvoir changer de modèle au cours du suivi peut être intéressant à plusieurs niveaux, et tout particulièrement dans un contexte de suivi à partir d'images aériennes. Notre principale motivation se place dans le cas où la caméra se rapproche de l'objet d'intérêt, partant d'une position très éloignée de cet objet. A ce moment-là, les images ne permettent pas d'extraire une quelconque information 3D de la scène. Le plus approprié est alors d'utiliser un modèle 3D simple ou même un modèle 2D plan pour représenter l'objet d'intérêt. Au fur et à mesure que la caméra se rapproche, la structure 3D devient plus visible ; il est possible de l'estimer tout en continuant le suivi. On peut trouver deux avantages à utiliser ce raisonnement : tout d'abord, la structure 3D que l'on extrait peut être utile pour d'autres applications que celui du suivi (l'estimation de la scène depuis un autre point de vue par exemple). D'autre part, le suivi classique (n'utilisant qu'un seul modèle) peut échouer lorsque la caméra s'approche de l'objet et que les changements d'échelles sont important alors que le fait de pouvoir changer de modèle permet de suivre l'objet d'intérêt dans ces conditions.

4.4.2 L'intégration de plusieurs modèles

L'avantage de l'approche statistique est que l'on a beaucoup de liberté pour modéliser et propager les particules. On tire profit de cela, et on en profite pour introduire différents modèles dans notre algorithme, comme des ellipses, cubes, pyramides, sphères .... Et voici comment on procède : on définit l'état de notre objet par

$$\mathbf{X} = (lab, \mathbf{x}), x \in \Re^N, lab \in \{1, ..., N\}$$

où lab est une variable discrète qui étiquette le modèle courant, et $\mathbf{x}$ est un vecteur de paramètres dans l'espace d'état qui décrit notre objet. Dans notre cas, les paramètres de notre objet sont la position 3D, la taille et l'orientation de l'objet. Il est important de souligner que les différents modèles n'ont pas nécéssairement les mêmes paramètres. Pour simplifier la représentation de nos modèles, on a choisi de prendre comme vecteur d'état l'union de tous les paramètes de tous les modèles et de mettre à 0 les paramètres inutilisés pour un modèle donné.

D'autre part, l'introduction de plusieurs modèles dans l'algorithme rend le processus de propagation des particules plus complexe. Ce processus ne peut plus se faire en une seule étape, mais doit être décomposé. On a choisi la décomposition suivante :

$$p(\mathbf{X}_t \mid \mathbf{X}_{t-1}) = p(\mathbf{x}_t \mid lab_t, \mathbf{X}_{t-1}) P(lab_t \mid \mathbf{X}_{t-1})$$

$P(lab_t \mid \mathbf{X}_{t-1})$ représente la probabilité de transition du modèle i au modèle j. Elle est définie par :

$$p(lab_t = j \mid \mathbf{x}_{t-1}, lab_{t-1} = i) = T_{ij}(\mathbf{x}_{t-1})$$

où T_{ij} sont les probabilités de transition.

Et $p(\mathbf{x}_t \mid lab_t, \mathbf{X}_{t-1})$ représente le processus de propagation des particules. La difficulté ici est que les paramètres d'état sont différents d'un modèle à un autre. Il est donc nécessaire de commencer par changer les paramètres d'état si le modèle change avant d'appliquer le processus

classique de propagation. On appelle h_{ij} la fonction qui modifie les paramètres d'état (si nécessaire) pour passer du modèle i au modèle j, et la probabilité $p_j(\mathbf{x}_t \mid h_{ij}(\mathbf{x}_{t-1}))$ correspondant au processus classique de propagation garantissant l'évolution de la particule. Pour illustrer le fontionnement de la fonction h_{ij}, considérons le cas simple où l'on souhaite passer du modèle *pyramide* au modèle *rectangle* : le vecteur d'état de la pyramide étant $(rect, x, y, z, L, l, 0)$, il suffit de mettre l'information de hauteur à zéro pour avoir un rectangle. La position du centre du modèle reste la même, et la taille du rectangle de base est inchangée elle aussi. Pour le passage d'un modèle à un auter dans des cas plus complexes, les détails sont donnés en Annexe B.

L'ensemble du processus peut finalement s'écrire de la façon suivante :

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, lab_{t-1} = i, lab_t = j) = p_j(\mathbf{x}_t \mid h_{ij}(\mathbf{x}_{t-1}))$$

Pour assurer la bonne évolution de l'ensemble des particules, on a besoin de définir une méthode de rééchantillonnage. Comme il a été dit précédemment, les méthodes existantes sont nombreuses, et on a choisi une méthode de rééchantillonnage systématique pour sa simplicité à mettre en œuvre, et parce que c'est la méthode la plus employée à l'heure actuelle. La procédure complète est décrite au paragraphe 3.3.4.

Le schéma global de l'algorithme avec les différents modèles est présenté dans la Figure 4.3.

Grâce à ce schéma de suivi, les transitions d'un modèle à un autre se font automatiquement et il est donc possible de passer d'un modèle à un autre au cours du suivi. Chaque état discret qui a un poids non nul contribue donc à la distribution finale de l'état de l'objet. Cela permet à plusieurs modèles d'être conservés en même temps lorsqu'il y a ambiguïté sur l'objet à suivre. Dès qu'un modèle fait une prédiction plus vraisemblable, ce modèle va dominer.

4.4.3 L'implémentation de l'algorithme

Ce paragraphe a pour objectif de décrire les choix que nous avons faits pour la description des objets, la façon de propager les particules, d'assigner les poids, et l'initialisation de l'algorithme. Une partie sera consacrée à l'estimation du mouvement de la caméra, étape annexe mais indispensable au bon fonctionnement de l'algorithme.

La représentation des objets

Dans notre méthode, nous représentons notre objet par un modèle 2D ou 3D constitué de segments. Un objet est donc défini par 9 paramètres, 6 pour la pose de l'objet dans l'espace (orientation et translation), et 3 paramètres pour sa taille (longueur, largeur et hauteur).

Notre base de données de modèles comprend :
- 2 modèles 2D : rectangle et ellipse
- 4 modèles 3D : parallélépipède, cylindre, pyramide et sphère

A partir des informations de longueur, largeur et hauteur, on construit nos modèles de la façon suivante : les modèles 2D n'utilisent que les informations de longueur L et largeur l (la hauteur est nulle). Leur centre est positionné en $(0, 0, 0)$, et chacun de leur sommets en $(\pm L/2, \pm l/2, 0)$. Quant aux modèles 3D, on traite le cas de la sphère à part. Les autres modèles sont centrés en $(0, 0, 0)$ comme les modèles 2D, et leurs sommets en $(\pm L/2, \pm l/2, 0\, ou\, h)$. La sphère quant à elle

Etant donné l'ensemble des particules $\{(\mathbf{x}_{t-1}^{(k)}, w_{t-1}^{(k)}), k = 1, ..., N\}$ à l'instant $t-1$, l'objectif est de construire le nouvel ensemble $\{(\mathbf{x}_t^{(k)}, w_t^{(k)}), k = 1, ..., N\}$ à l'instant t.

Pour chaque particule k de l'ensemble :

1. On prédit par échantillonnage l'évolution de la particule
   - On calcule les probabilités de transition pour trouver le label du modèle $lab_t^{(k)}$, et on calcule les nouveaux paramètres d'état de la particule si le modèle a changé
   $$P(lab_t^{(k)} = j \mid \mathbf{X}_{t-1}^{(k)} = \mathbf{x}_{t-1}^{(k)}) = T_{ij}(\mathbf{x}_{t-1}^{(k)})$$
   - On propage la particule selon le modèle dynamique du système pour obtenir $\mathbf{x}_t^{(k)}$
   $$p(\mathbf{x}_t^{(k)} \mid \mathbf{X}_{t-1}^{(k)}, lab_t^{(k)} = j) = p_j(\mathbf{x}_t^{(k)} \mid h_{ij}(\mathbf{x}_{t-1}^{(k)}))$$
2. On mesure et pondère le nouvel état de la particule selon les observations effectuées $\mathbf{Y}_t$
   $$w_t^{(k)} = p(\mathbf{Y}_t \mid \mathbf{X}_t = \mathbf{x}_t^{(k)})$$
   et on normalise les poids pour que
   $$\sum_{k=1}^{N} w_t^{(k)} = 1$$
3. On fait l'étape de rééchantillonnage pour assurer la concentration des particules dans les régions d'intérêt.

FIG. 4.3 – L'algorithme de suivi basé sur des modèles géométriques.

est centrée en $(0, 0, h/2)$, de manière à ce qu'elle soit posée sur le plan $z = 0$. Les différentes représentations des modèles sont données sur la figure 4.4.

Concernant la détermination des probabilités pour passer d'un modèle à un autre, on suppose que $T_{ij}(\mathbf{x}_t) \equiv T_{ij}$, et on spécifie ces probabilités à la main. La matrice de transition utilisée pour toutes nos expériences est la même :

$$T = \begin{pmatrix} 0.9 & 0.028 & 0.028 & 0.014 & 0.014 & 0.014 \\ 0.028 & 0.9 & 0.014 & 0.028 & 0.014 & 0.014 \\ 0.028 & 0.014 & 0.9 & 0.028 & 0.014 & 0.014 \\ 0.014 & 0.028 & 0.028 & 0.9 & 0.014 & 0.014 \\ 0.028 & 0.014 & 0.028 & 0.014 & 0.9 & 0.014 \\ 0.014 & 0.028 & 0.014 & 0.028 & 0.014 & 0.9 \end{pmatrix}$$

Dans l'idéal, il faudrait trouver un moyen pour fixer les paramètres de cette matrice de transition automatiquement. Une méthode pourrait être basée sur des techniques d'apprentissage ; nous reviendrons sur ce point dans le dernier chapitre de la thèse.

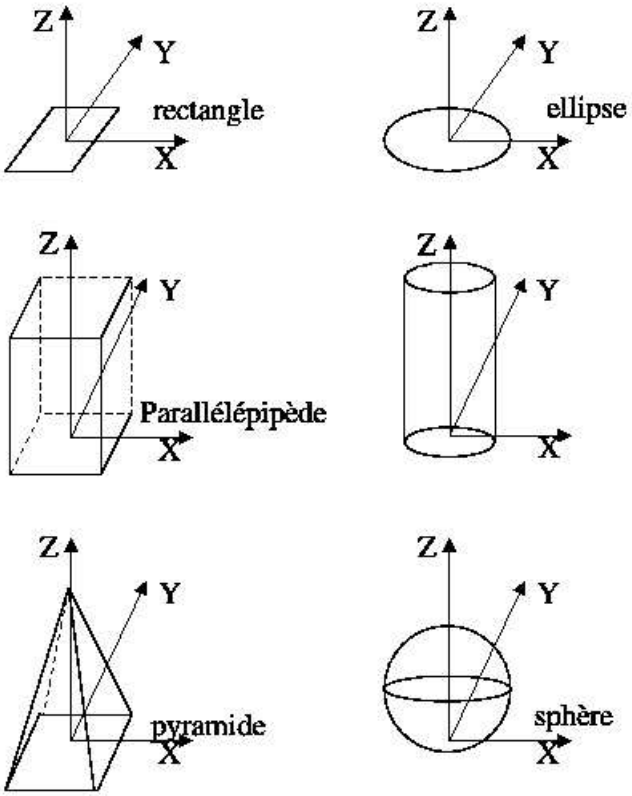

FIG. 4.4 – La base de donnée des modèles

Le modèle dynamique

Le modèle dynamique du système est simple : on s'intéresse au suivi d'objets statiques avec une caméra en mouvement, donc le modèle de mouvement de l'objet est réduit à $\mathbf{X}_t = \mathbf{X}_{t-1} + \mathbf{b}_t$ où $\mathbf{b}_t$ est un bruit aléatoire gaussien.

Le bruit en position, qui permet la bonne exploration de l'image, est fixé automatiquement. On choisit le rayon d'un disque dans l'image (donc en nombre de pixels) ; ce disque correspond à la surface que l'on veut couvrir par les particules. Par projection d'un modèle dans une image on est capable, pour toute image, de déterminer le rayon 3D correspondant au rayon du disque choisi dans l'image. Nous évaluerons la robustesse de l'algorithme par rapport à ce paramètre dans le chapitre suivant.

Le bruit en taille est exprimé en pourcentage de la taille courante des modèles. Nous évaluerons la robustesse de l'algorithme par rapport à ce paramètre dans le chapitre suivant.

Comme nous venons de l'expliquer, notre modèle de mouvement est très simple en raison de nos hypothèses. Dans le cas ou l'objectif serait de suivre des objets en mouvement, il est facile de modifier le modèle de mouvement pour introduire et prendre en compte la vitesse de l'objet par exemple, ou son accélération ; on peut se référer à [IB98, JSR05] pour plus de détails.

L'affichage du résultat

A chaque instant, on doit afficher un résultat pour représenter la position estimée du suivi. On pourrait facilement calculer cette estimée en prenant la moyenne pondérée de l'ensemble des particules, comme on le fait classiquement lorsque l'on a un unique modèle([IB96b] par exemple) : mais cela n'a pas de sens. On peut plutôt penser que les différents états de l'objet correspondent à des types différents, et on a alors besoin d'une approche à plusieurs étapes pour calculer la meilleure estimée. Par conséquent, on estime d'abord le type de modèle pour l'objet en utilisant le MAP (Maximum A Posteriori) comme suit :

$$\begin{aligned} \hat{lab}_t &= argmax_j P(lab_t = j \mid \mathbf{Y}_1 \dots \mathbf{Y}_t) \\ &= argmax_j \sum_{k \in \Gamma_j} w_t^{(k)} \end{aligned}$$

où $\Gamma_j = \{k \mid \mathbf{X}_t^{(k)} = (j, \mathbf{x}_t^{k)})\}$.

Puis, on obtient l'estimée pour le vecteur de paramètres d'état en déterminant la moyenne pondérée des particules correpondantes au modèle déterminé à l'étape précédente :

$$\hat{\mathbf{x}}_t = \frac{\sum_{k \in \hat{\Gamma}} w_t^{(k)} \mathbf{x}_t^{(k)}}{\sum_{k \in \hat{\Gamma}} w_t^{(k)}}$$

où $\hat{\Gamma} = \{k \mid \mathbf{X}_t^{(k)} = (\hat{lab}_t, \mathbf{x}_t^{k)})\}$. C'est cette estimée que l'on utilise pour afficher nos résultats sur les figures de ce document.

L'initialisation

L'initialisation de notre algorithme est quasi automatique : on donne une position approximative de notre objet dans la scène pour construire notre premier ensemble de particules. On n'a pas besoin de connaître le type d'objet que l'on veut suivre, l'algorithme construit l'ensemble initial des particules en générant des modèles de chacun des types. Les poids associés vont décider ensuite quel est le modèle le plus approprié.

Dans les expériences effectuées pour valider notre algorithme, nous avons aussi évalué la robustesse de l'algorithme par rapport à une initialisation peu précise, concernant la position ou la taille de la cible que l'on souhaite suivre dans la scène. Les résultats de cette évaluation sont présentés dans le chapitre suivant.

Les poids des particules

Il reste à décrire comment on attribue les poids aux particules. Le modèle de l'objet est un modèle 2D ou 3D et le modèle d'observation est l'image courante ; il n'est donc pas possible de calculer la vraisemblance d'un modèle directement. On a choisi de comparer leurs contours dans l'image. Pour cela, on reprojete notre modèle (2D ou 3D) dans l'image courante et on calcule la distance entre les contours du modèle reprojeté et ceux de l'image. Pour cela on utilise une *carte de Chanfrein* : c'est une image de distances qui fournit pour chaque pixel de l'image sa distance au plus proche point de contour dans l'image.

Les distances de Chanfrein sont des distances discrètes, utilisées en analyse d'images, permettant d'approcher au mieux les distances euclidiennes. Ces distances reposent sur la définition et l'application de masques de pondération. Les algorithmes de base sont dus à Rosenfeld [RP66], et leur grande efficacité est à l'origine des développements des distances de Chanfrein, mises au point par Borgefors [Bor84, Bor86].

On utilise un détecteur de Canny-Deriche pour extraire les contours de l'image, et on cherche à comparer ces contours à ceux des modèles (2D ou 3D) que l'on reprojette dans l'image. En ce qui concerne la reprojection des modèles, notons que l'on ne reprojette que les contours visibles ; on projette tout d'abord les sommets puis on trace les arêtes joignant les différents sommets. Il faut ensuite attribuer une note à chaque particule, évaluant la similarité entre les contours du modèle reprojeté et les contours extraits dans l'image. Pour cela, nous avons formulé les critères suivants :

- Plus le modèle est grand (un grand modèle est un modèle avec beaucoup de points de contour), plus la note doit être élevée
- Plus la somme des distances entre les points de contour du modèle et les points de contour extraits de l'image est faible, plus la note doit être élevée

Nous avons donc fixé une première note de la manière suivante :

$$Note_1 = \frac{N_c}{\sum_i d_i}$$

où N_c représente le nombre de points de contour du modèle reprojeté, et d_i les distances des points de contour du modèle aux points de contour de l'image.

FIG. 4.5 – Contours de l'image et carte de distance (haut) ; un modèle 3D reprojeté (en bas à gauche) et surimposé à l'image originale (en bas à droite).

Le problème avec cette note est la tendance à sélectionner les petits modèles. En effet, ces petits modèles ont bien souvent beaucoup moins de points de contours, mais aussi des distances beaucoup plus petites. De plus, ils parviennent beaucoup plus facilement à se « caler » pour coller aux contours extraits dans les images.

Nous avons donc essayé d'autres notes pour essayer de lutter contre la préférence de l'algorithme pour les petits modèles :

$$Note_2 = \frac{N_c^2}{\sum_i d_i} \quad Note_3 = \frac{N_c^{3/2}}{\sum_i d_i} \quad ou \quad Note_4 = \frac{N_c^2}{(\sum_i d_i)^{3/2}}$$

Mais il apparaît difficile de trouver LA note qui correspond bien à ce que l'on veut. En effet, on peut considérer le problème du choix de la note comme un problème de sélection de modèle : N_c représente la complexité du modèle, et $\sum_i d_i$ le terme de fidélité aux données. Il s'agit de trouver le meilleur compromis entre complexité du modèle et fidélité aux données ; cependant, ce problème est plus complexe que celui de la sélection du nombre de classes de l'histogramme.

Cette remarque initiale et ces différents tests nous ont amené à trouver un moyen de pondérer les modèles pour éviter ce phénomène. Pour une taille donnée (longueur, largeur, hauteur), et pour une orientation fixée, nous avons regardé le nombre de points de contour des différents modèles reprojetés dans une image. Nous avons pu en extraire des coefficients de pondération, qui sont :

$$c_{rect} = 1.0 \quad c_{ell} = 1.12 \quad c_{parall} = 1.70 \quad c_{cyl} = 1.90 \quad c_{pyr} = 1.35 \quad c_{sph} = 0.78$$

Nous avons effectué nos expérimentations avec la note $Note_3$ pondérée par ces coefficients.

L'estimation du mouvement de la caméra

Jusqu'à présent nous avons parlé de modèle à reprojeter dans les images mais nous n'avons pas encore expliqué comment nous procédons pour cela. En considérant le vecteur d'état suivant

$$\mathbf{X} = \{lab, x, y, z, L, l, h, \theta_x, \theta_y, \theta_z\}$$

où x, y et z sont les coordonnées du centre du modèle, L, l et h ses dimensions et θ_x, θ_y et θ_z son orientation, on constate qu'il est possible de générer des particules en faisant varier tous les

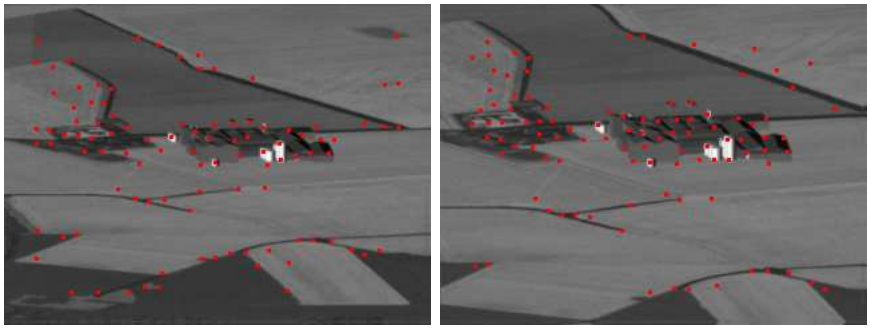

FIG. 4.6 – L'extraction des points d'intérêt ; les points ne sont pas extraits de manière uniforme dans toute la scène.

paramètres. Mais pour avoir une chance que ca marche, il faudrait générer un nombre beaucoup trop important de particules, et c'est irréalisable en pratique. L'idée est donc de diminuer le nombre de paramètres dans le vecteur d'état.

Notre idée au départ était d'élaborer un algorithme de suivi en utilisant le moins d'information possible sur la caméra, la position du porteur ou celle de la cible dans l'image. A partir de l'unique connaissance de la focale de la caméra, et avec l'hypothèse que la scène considérée est plane, il est théoriquement possible de remonter aux matrices de projection, nécéssaires pour projeter nos modèles (2D ou 3D) correctement dans nos images. L'algorithme consiste à estimer le plan dominant dans les images, puis de remonter aux rotations et translations relatives par rapport aux images précédentes. L'algorithme est décrit au chapitre 1.

En pratique, nous avons eu de nombreuses difficultés pour estimer le mouvement à partir des images de nos séquences. Le bon fonctionnement de l'algorithme repose sur l'extraction des points d'intérêt dans la scène de manière uniforme ; avec les séquences de synthèse que l'on avait à disposition (fournies par Thalès Optronique), les points d'intérêt extraits étaient mal répartis dans la scène. L'estimation du plan dominant était mauvaise, et nous n'avons pas pu remonter au mouvement de la caméra de manière suffisamment précise. La figure 4.6 met en évidence le problème de l'homogénéisation des points d'intérêt dans différentes images.

Lorsque la scène est déjà 3D (c'est-à-dire que la caméra est suffisamment proche de la scène pour avoir des informations 3D réelles des objets de la scène), l'estimation du plan du sol reste délicate parce que les points d'intérêt sont extraits à la fois sur le plan du sol, et sur les objets 3D. Dans ce cas, l'algorithme basé sur l'estimation du plan dominant ne fonctionne pas toujours bien. Nous avons donc testé des approches plus classiques, basées sur la géométrie épipolaire, l'estimation de la matrice essentielle pour remonter au mouvement de la caméra. Mais là encore, les méthodes classiques ont échoué.

En réalité le principal problème est que l'estimation du mouvement pour une scène plane et une caméra orthographique est dégénéré. Dans notre cas, la caméra est proche d'une caméra orthographique ; l'estimation du mouvement est donc instable. En effet, le rapport entre la taille des objets et la distance de la caméra à la scène est très inférieur à 0.1. Dans ce cas, le modèle perspectif devient une sur-paramétrisation et c'est le modèle orthographique qui est adapté.

Ce problème n'étant pas central pour nous, nous avons décidé d'utiliser les données associées aux séquences fournies par Thalès pour remonter aux matrices de projection dont nous avons besoin pour notre algorithme. Les positions relatives du porteur et de la cible données dans un repère commun, et la connaissance de la focale de la caméra, nous permettent de remonter aux matrices de projection. L'algorithme est donné en Annexe A.

4.5 Un algorithme de suivi combinant des modèles géométriques et des histogrammes de couleur

4.5.1 La problématique

Comme nous l'avons dit précédemment, l'utilisation d'histogrammes de niveaux de gris s'avère efficace pour effectuer un suivi d'objet dans une séquence d'images, mais ne permet en aucun cas de remonter à une quelconque information 3D réelle de la scène. L'utilisation du mouvement de la caméra pour reprojeter des modèles géométriques simples dans la scène 3D permet quant à elle d'extraire et d'utiliser de l'information 3D. Sur les séquences de synthèse où la segmentation ne pose pas de problèmes particuliers, l'algorithme est performant. Mais les expérimentations qui ont été faites sur des séquences rélles ont montré les faiblesses de l'« appariement de contours ». En effet, lorsque la densité de contours est trop élevée, les modèles se raccrochent facilement à d'autres contours, et le suivi échoue dans de nombreux cas. Nous invitons le lecteur à se reporter au chapitre suivant pour les résultats des expériementations.

Nous avons donc décidé de combiner les deux approches développées précédemment pour obtenir un algorithme performant de suivi d'objets à partir de séquences d'images, dans le cadre particulier des images aériennes.

4.5.2 L'algorithme

Ce paragraphe a pour objectif de présenter les choix qui ont été faits pour l'implémentation de l'algorithme : initialisation, changements de modèle, mise à jour de l'histogramme .... La plupart du temps, nous passerons rapidement sur ces choix étant donné qu'ils ont été décrits de manière détaillée dans les deux sections précédentes.

L'initialisation

Pour l'initialisation, on procède de la manière suivante : un ensemble de particules est généré comme pour l'algorithme utilisant les modèles géométriques (se référer au paragraphe 4.4.3). La meilleure particule au sens de la comparaison des contours extraits de l'image et de ceux d'un modèle géométrique reprojeté sert d'initialisation pour les histogrammes. Nous utilisons, de même que pour l'algorithme utilisant les histogrammes, le critère de sélection automatique du nombre de classes de l'histogramme (voir paragraphe 4.3.3).

La représentation des objets, le modèle dynamique et l'affichage du résultat

C'est la même que pour l'algorithme utilisant les modèles géométriques (paragraphe 4.4.3).

Le poids des particules

Dans cet algorithme, il faut combiner une note liée au bon appariement des contours extraits de l'image et ceux des modèles géométriques reprojetés dans l'image, et une note liée à l'appariement des histogrammes de niveaux de gris du modèle et des différentes particules.

La note utilisée pour l'appariement des contours est la même que celle utilisée pour l'algorithme utilisant les modèles géométriques. Elle est décrite au paragraphe 4.4.3 et dépend du nombre de points de contour du modèle et de la somme des distances entre les points de contour du modèle et les points de contour extraits de l'image. Elle est pondérée par un coefficient lié au modèle géométrique (voir le paragraphe 4.4.3).

La note utilisée pour la comparaison d'histogrammes est la même que celle utilisée pour l'algorithme utilisant les histogrammes. Elle est basée sur le coefficient de Bhattacharyya et est décrite au paragraphe 4.3.2.

La note finale est obtenue par multiplication des deux notes ; la note reliée aux histogrammes est comprise entre 0 et 1. On peut donc voir la note finale comme la note liée aux contours pondérée selon la « bonne allure » de l'histogramme.

La mise à jour de l'histogramme

Elle se fait à intervalles réguliers, toutes les N_h images. Dans l'évaluation, nous montrerons l'influence du nombre N_h sur les performances de l'algorithme. En théorie, il serait bon que l'histogramme se mette à jour uniquement lorsqu'il en a besoin et non pas de manière régulière.

Le rééchantillonnage des particules

Il se fait de la même manière que pour l'algorithme de suivi utilisant des modèles géométriques (paragraphe 4.4.2).

4.6 Conclusions

Nous avons présenté dans ce chapitre trois algorithmes de suivi d'objets à partir d'une séquence d'images acquise par une caméra. Le premier algorithme repose sur l'utilisation d'histogrammes de niveaux de gris combinés à un filtrage particulaire, le deuxième sur une comparaison des contours de modèles géométriques reprojetés dans la scène aux contours extraits des images, et le dernier intègre à la fois les informations de contours et celles de couleur pour rendre l'algorithme plus robuste.

L'objectif des chapitres suivants est de valider expérimentalement ces algorithmes, aussi bien sur des séquences acquises par une caméra standard que par une caméra embarquée à bord d'un avion. Ce cadre spécifique des images aériennes, dans lequel les difficultés rencontrées sont souvents importantes, est l'application importante de ces travaux de thèse.

Chapitre 5

Validation de l'algorithme de suivi basé sur des histogrammes de couleurs

Sommaire

5.1 Introduction

L'objectif de ce chapitre est de présenter l'évaluation pour notre algorithme de filtrage particulaire basé sur des histogrammes de couleurs. Tout d'abord, nous mettons en évidence l'apport de la sélection automatique du nombre de classes et de la division de l'ellipse en quartiers, puis nous présentons les résultats expérimentaux de notre algorithme testé sur trois séquences très différentes par le type d'objet à suivre et par le type de difficultés rencontrées dans le suivi (en termes de changement d'échelle, vitesses des objets dans la scène, caméra statique ou en mouvement...). Enfin, des idées n'ayant pas pu être mises en œuvre seront évoquées dans un paragraphe : cela clorera le chapitre.

5.2 Evaluation des contributions

L'objectif de cette partie est de prouver que les critères mis en place pour la sélection automatique du nombre de classes des histogrammes et pour prendre en compte l'information spatiale (perdue par l'utilisation des histogrammes) améliorent les performances du suivi. Nous avons mis en place des expérimentations pour valider nos contributions.

5.2.1 La description des séquences

Nous avons utilisé trois séquences pour valider notre algorithme, une image de chaque séquence est représentée sur la figure 5.1.

1. *Le suivi de visage* : La séquence présente une personne entrant et bougeant dans une pièce (http ://www.ee.oulu.fi/~mikak/tracking/FaceColor.html). C'est une séquence de 500 images de 384x288 pixels. La taille des demi axes de l'ellipse représentant l'état de l'objet d'intérêt (position et taille dans l'image, voir le paragraphe 4.3.2 page 63) dans la première image est de 12 et 18 pixels.

2. *Le suivi de voiture* : C'est une des séquences de tests de la base de l'atelier PETS [2] (ftp ://pets2001.cs.rdg.ac.uk/) 2001 dans le contexte d'une application d'assistance à la conduite. La séquence contient 500 images.

3. *Le suivi de joueur de football* : La séquence est tirée d'un match de football, et contient 200 images de 889x676 pixels. La taille des demi axes de l'ellipse dans la première image est de 26 et 15 pixels.

FIG. 5.1 – Une image de chacune des trois séquences.

Pour juger de la réussite ou de l'échec du suivi, nous avons établi une vérité terrain pour chacune des séquences, en créant une application permettant de conserver dans un fichier les coordonnées du centre de l'objet d'intérêt pour chaque image d'une séquence.

5.2.2 Validation du critère de sélection automatique du nombre de classes des histogrammes

Nous avons exécuté notre algorithme plusieurs fois avec la sélection automatique du nombre de classes et pour un nombre de classes fixé, dans le cadre du suivi effectué avec l'ellipse entière. Nous utilisons l'état moyen pour la présentation des résultats (voir le paragraphe 4.3.2).

Le Tableau 5.1 fait la synthèse des résultats obtenus : le nombre indique le numéro de l'image pour laquelle le suivi échoue, et les nombres en gras correspondent aux résultats obtenus pour la sélection automatique du nombre de classes. Pour juger de la réussite ou de l'échec du suivi, nous avons utilisé la vérité terrain de la manière suivante : la détection d'un échec dans le suivi de l'objet se fait lorsque, dans une image, la taille de l'ellipse est trop grande ou le centre de

[2] Performance Evaluation of Tracking Systems

# de classes	Football	Indoor	Plaque
2	16	19	31
4	87	23	46
6	107	111	55
8	109	125	41
10	95	172	60
12	118	169	94
14	90	**168**	92
16	121	164	84
18	**117**	132	**93**
20	111	217	61
22	75	108	78
24	88	135	62
26	72	90	63
28	86	107	73
30	76	78	74
32	69	85	82
34	102	70	61
36	92	92	86
38	102	76	69
40	87	59	91

TAB. 5.1 – Performances de l'algorithme pour le suivi de l'ellipse entière avec différents nombres de classes.

l'ellipse trop éloigné de la vérité terrain (lorsque la distance entre le centre de l'ellipse et la vérité terrain est supérieure à 50, ou que l'ellipse est dix fois plus grande que l'ellipse initiale).

La première remarque que nous pouvons faire en regardant le tableau est que le résultat est assez sensible au nombre de classes. Puis on constate que le choix automatique du nombre de classes ne donne pas forcément le meilleur résultat ; mais dans tous les cas, il est proche du meilleur.

La conclusion de cette expérience est que fixer arbitrairement un nombre de classes n'est pas une bonne solution pour suivre des objets divers. Il est préférable d'utiliser le critère de sélection automatique du nombre de classes et avoir un résultat proche du meilleur que de fixer un nombre de classes de manière aléatoire et risquer un mauvais suivi.

5.2.3 Validation de la division de l'ellipse en quartiers

Nous avons effectué deux suivis pour chacune des séquences avec le critère de sélection automatique du nombre de classes : le premier utilisant l'ellipse entière et le second utilisant la division de l'ellipse en quartiers.

Les résultats sont présentés sous forme de courbes dans la figure 5.2. Sur chaque courbe, l'axe des abscisses représente le numéro des images, et l'axe des ordonnées l'erreur en position du centre de la particule représentant l'état moyen par rapport à la verité terrain (voir le paragraphe

5.2.1). Nous avons, pour chaque séquence, lancé l'algorithme plusieurs fois. Les résultats présentés sont ceux correspondant à une seule exécution, mais ils sont semblables pour l'ensemble des tests effectués.

Ces courbes montrent que l'utilisation des 4 quartiers améliore beaucoup les performances du suivi.

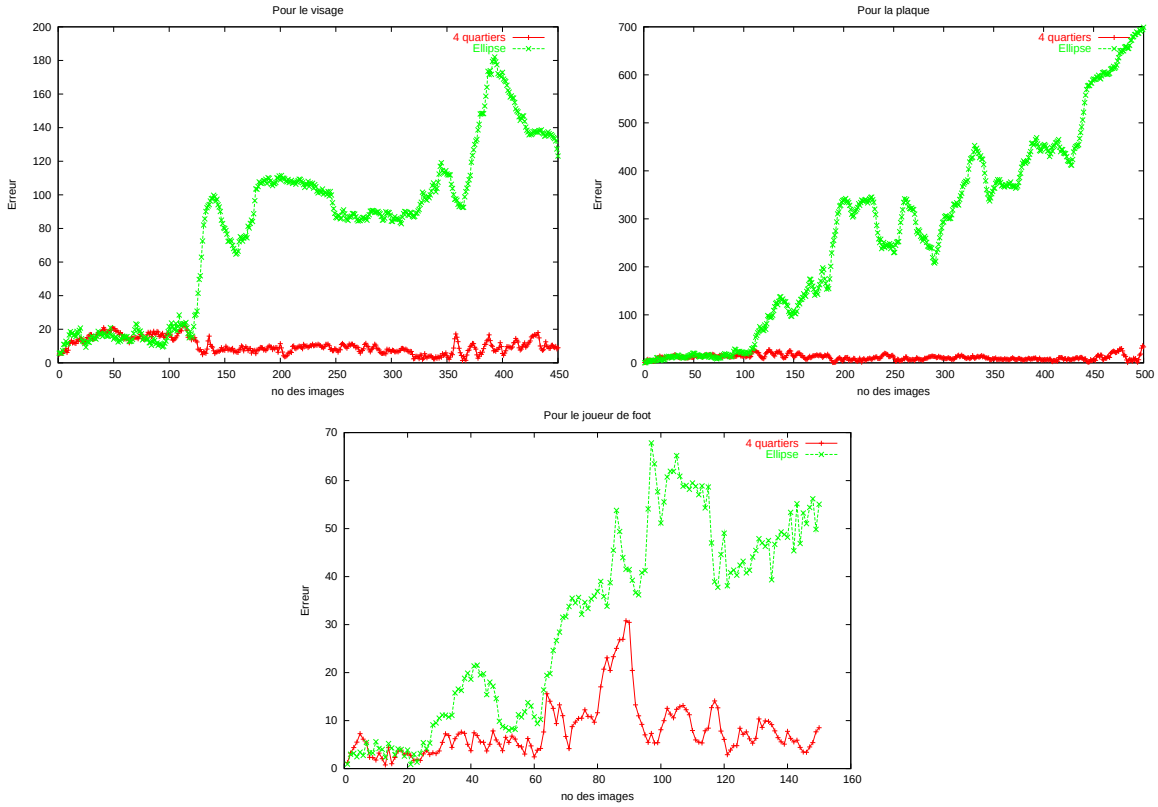

FIG. 5.2 – Influence de la division de l'ellipse en quartiers pour chacune des 3 séquences : la séquence du visage (en haut à gauche), de la plaque (en haut à droite) et du joueur de foot (en bas)

5.3 Résultats obtenus sur les séquences

Nous avons testé notre algorithme de suivi sur trois séquences : les résultats montrent que notre approche permet de l'utiliser pour des applications diverses, sans avoir besoin d'initialiser les paramètres du suivi à chaque fois. Trois ensembles de résultats sont fournis ci-dessous ; on utilise des séquences en niveaux de gris et la partition de l'ellipse en quartiers pour l'ensemble de nos expérimentations.

1. *Le suivi de visage* : La séquence présente une personne entrant et bougeant dans une pièce. Il y a de nombreux changements d'apparence à prendre en compte, la personne se retournant complètement au cours de la séquence. Les résultats montrent que la mise à jour du

FIG. 5.3 – Résultats du suivi de visage : images 1, 140, 395, 412 et 449 de la séquence.

modèle est performante et permet de prendre en compte les changements d'apparence de l'objet. D'autre part, l'algorithme est capable de suivre la personne, même si celle-ci se déplace à des allures différentes au cours de la séquence.

2. *Le suivi de voiture* : La séquence se place dans le contexte d'une application d'assistance à la conduite. Nous avons fait deux types de tests, les résultats sont présentés sur les Figures 5.4 et 5.5. Les difficultés de la séquence sont : la capacité à prendre en compte les mouvements rapides de l'objet tout autant que ceux de la caméra, les différents points de vue que l'on a de l'objet à suivre, des changements d'échelle et les rotations de l'objet d'intérêt dans le plan de l'image. Les résultats montrent que l'algorithme est performant tout au long de la séquence.

   – Pour la première expérimentation, nous avons suivi la voiture entière, et nous avons obtenu des résultats équivalents à ceux obtenus par [NKMG03]. Les demi axes de l'ellipse dans la première image sont de 40 et 50 pixels.

   – Pour la seconde expérimentation, nous avons suivi la plaque d'immatriculation de la voiture uniquement. Les demi axes de l'ellipse dans la première image sont de 12 et 18 pixels. La difficulté supplémentaire réside donc dans la taille de l'objet suivi. Les résultats présentés en Figure 5.5 montrent que notre algorithme est capable de suivre efficacement des objets même de petite taille.

3. *Le suivi de joueur de football* : La séquence est tirée d'un match de football ; les difficultés de cette séquence sont les mouvements rapides des joueurs et les occultations de certains joueurs par d'autres joueurs sur le terrain. Dans l'image 78, un joueur est en train de tomber et un autre tente d'attrapper le ballon, ce qui provoque une occultation importante du joueur. Notre algorithme reste performant dans ce cas.

FIG. 5.4 – Résultats du suivi de voiture : images 1, 187, 347, 462 et 563 de la séquence.

FIG. 5.5 – Résultat du suivi de la plaque : images 1, 187, 347, 417, 520 de la séquence, et un zoom fait sur l'image 417.

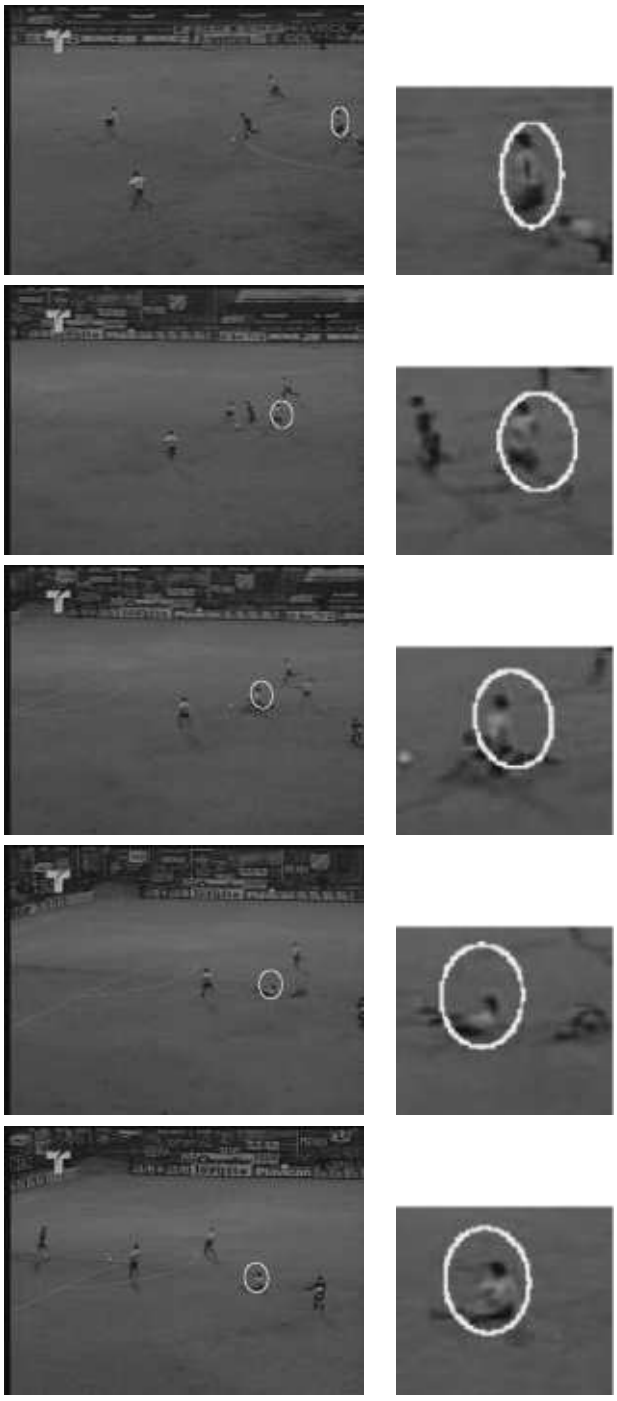

FIG. 5.6 – Résultats obtenus pour le suivi de joueur de football : images 1, 55, 76, 97 et 141 de la séquence. Dans l'image 76, le joueur blanc occulte le noir.

Les résultats présentés sur les figures ci-dessus sont ceux obtenus pour une exécution. Des résultats semblables ont été obtenus pour vingt exécutions par séquence ; les résultats présentés sont donc représentatifs.

5.4 Application au suivi de structures à partir d'images aériennes

Notre travail se place dans un cadre militaire, dans lequel l'objectif est d'être capable de suivre efficacement des objets à partir d'images aériennes. Nous avons donc testé notre algorithme dans différentes configurations. Sur des cas simples pour commencer : nous avons testé l'algorithme sur des séquences de synthèse fournies par Thalès Optronique. Puis dans un deuxième temps, nous l'avons testé sur des séquences réelles, dans lesquelles les difficultés sont beaucoup plus nombreuses.

5.4.1 Séquences de synthèse

Nous avons deux séquences de synthèse :

A basse altitude

La séquence, filmée à basse altitude, est composée de 1200 images. L'objectif est de suivre le cylindre situé au centre de la scène. La trajectoire de l'avion est la suivante : tout d'abord la caméra se rapproche de la cible, puis elle tourne autour avant de s'en éloigner.

Les résultats du suivi sont présentés sur la figure 5.7. L'évaluation a été faite avec 100 particules. L'objet d'intérêt est bien contrasté par rapport au reste de la scène. L'algorithme n'a pas de mal à le suivre efficacement.

A moyenne altitude

La séquence se compose de 1400 images de la même scène que précédemment. La seule différence se situe dans l'altitude de l'avion qui est beaucoup plus haute. Les objets dans la scène sont donc plus petits, et il est plus difficile de les suivre de manière efficace.

Les résultats du suivi sont présentés sur la figure 5.8. L'évaluation, comme précédemment a été effectuée avec 100 particules. Dans cette séquence, l'objet d'intérêt est moins contrasté que dans la précédente, ce qui rend le suivi plus difficile. Cette observation se retrouve dans les résultats obtenus : l'ellipse a tendance à grossir et à englober du fond de la scène. Le suivi est moins bon que pour la séquence à basse altitude.

En réalité, ce qui se passe dans cette séquence n'est pas surprenant : l'ellipse initiale englobe non seulement l'objet d'intérêt, mais aussi une zone du fond, qui est sur cette séquence une zone homogène. Etant donné que nous attribuons des poids aux pixels en fontion de leur éloignement du centre de l'ellipse, les pixels du bord ne contribuent que peu à la note. Le grossissement de l'ellipse est donc peu pénalisée lorsqu'en dehors de l'ellipse initiale, les niveaux de gris sont les mêmes qu'à l'intérieur. Une amélioration possible à l'algorithme serait de détecter cela à l'initialisation.

5.4.2 Séquences réelles

Ces séquences nous ont été fournies par Thalès Optronique.

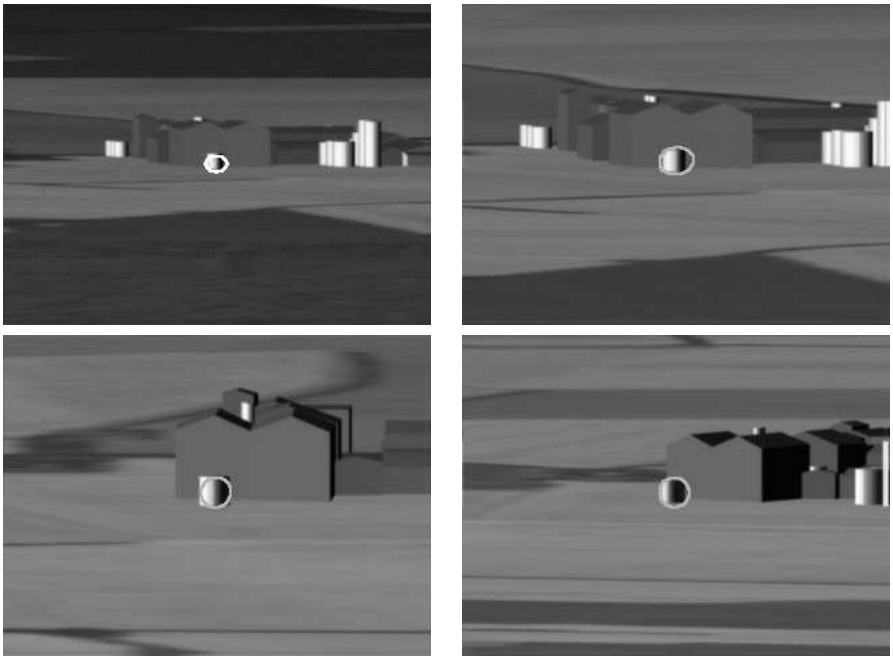

FIG. 5.7 – Suivi sur la séquence à basse altitude : la particule au meilleur score est représentée (images 0, 400, 800 et 1200).

Séquence vol1 p34

Le scénario de la séquence est le suivant : l'avion se rapproche progressivement d'un champ bordé d'arbres dans lequel quatre véhicules sont stationnés. La trajectoire de l'avion est une trajectoire d'approche, suivie, à la fin de la séquence, d'une « phase de break » , phase durant laquelle l'avion rentre dans un virage pour effectuer une rotation importante.

La séquence comporte 1100 images. L'objectif est de suivre un des quatre véhicules stationnés. Au départ, le véhicule n'est représenté que par quelques pixels, mais au fur et à mesure, la résolution augmente et le véhicule est très détaillé.

Les résultats du suivi sont présentés sur la figure 5.9. Au vu des résultats, nous pouvons faire les remarques suivantes. Même avec une bonne initialisation, l'algorithme ne parvient pas à suivre l'objet d'intérêt de manière efficace. Dès l'image 300, l'ellipse contenant notre objet est trop grande, et beaucoup du fond de la scène est alors pris en compte pour le calcul de l'histogramme. Si la proportion de fond est trop grande, l'algorithme n'est plus capable de distinguer l'objet du reste de la scène et ne peut plus « suivre » l'objet d'intérêt. En modifiant le nombre de particules ou les valeurs de bruit, les résultats ne sont pas vraiment meilleurs.

Une explication possible est le « changement de luminosité » de la scène au cours du temps. En effet, même si les niveaux de gris du véhicule sont toujours élevés (et proche de la saturation), les niveaux de gris du reste de la scène sont beaucoup modifiés. A l'image 600, les niveaux de gris du pré aux environs du véhicule sont compris entre 162 et 170, alors qu'à l'image 900, ils sont compris entre 152 et 162. Puisque la distance entre les histogrammes n'est pas invariante à des changements de luminosité, cela pose donc des problèmes. D'autre part, l'ombre du véhicule est

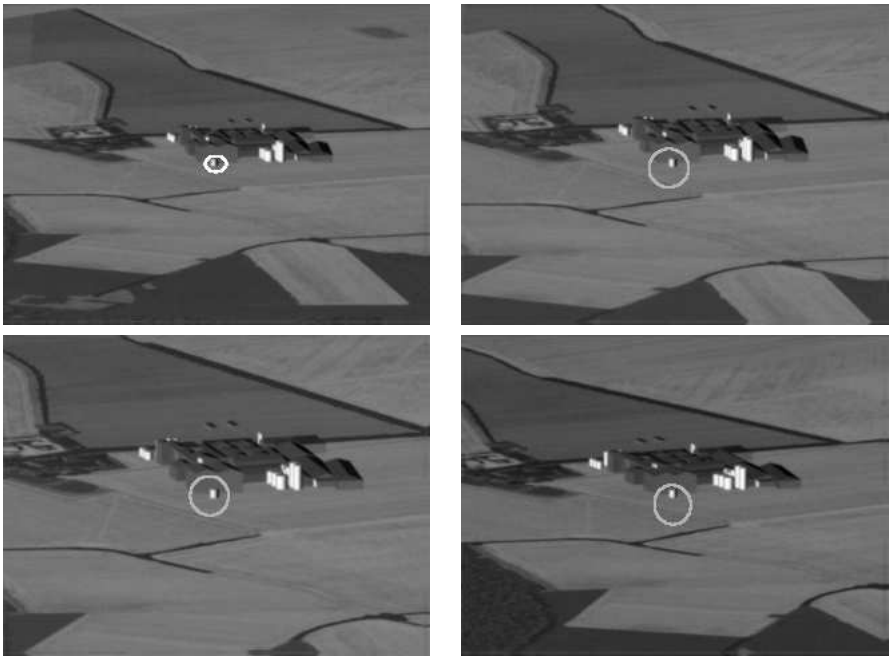

FIG. 5.8 – Suivi sur la séquence à moyenne altitude : la particule au meilleur score est représentée (images 0, 400, 800 et 1300).

beaucoup plus grande à l'image 900 qu'à l'image 300 : cela rajoute une difficulté supplémentaire pour le suivi, et peut être à l'origine d'une mauvaise mise à jour du modèle.

Séquence vol0 p3

Le scénario de la séquence est le suivant : l'avion se rapproche puis tourne autour d'un véhicule stationné à côté d'une petite route.

La séquence, constituée d'images infrarouges, comporte 480 images. Les résultats du suivi sont présentés sur la figure 5.10.

Comme pour la séquence précédente, les résultats du suivi sont plutôt mauvais. En augmentant le nombre de particules ou en modifiant les paramètres de bruit, les résultats ne sont pas meilleurs. Ici aussi, nous constatons qu'un changement de luminosité de la scène est une explication possible à l'échec de l'algorithme. La différence est que dans cette séquence, les niveaux de gris du véhicule sont faibles au début de la séquence (il est difficile même à l'œil nu de le distinguer par ses niveaux de gris par rapport au reste de la scène), alors qu'à partir de l'image 300, le moteur chaud du véhicule rend les niveaux de gris de la partie avant très élevés (proche de la saturation). Le reste de la scène, dont les niveaux de gris sont compris entre 80 et105 au début de la séquence et entre 65 et 95 à l'image 300, est beaucoup modifié également. Ce changement de luminosité au cours de la séquence est probablement à l'origine des mauvaises performances de l'algorithme sur ces deux séquences réelles.

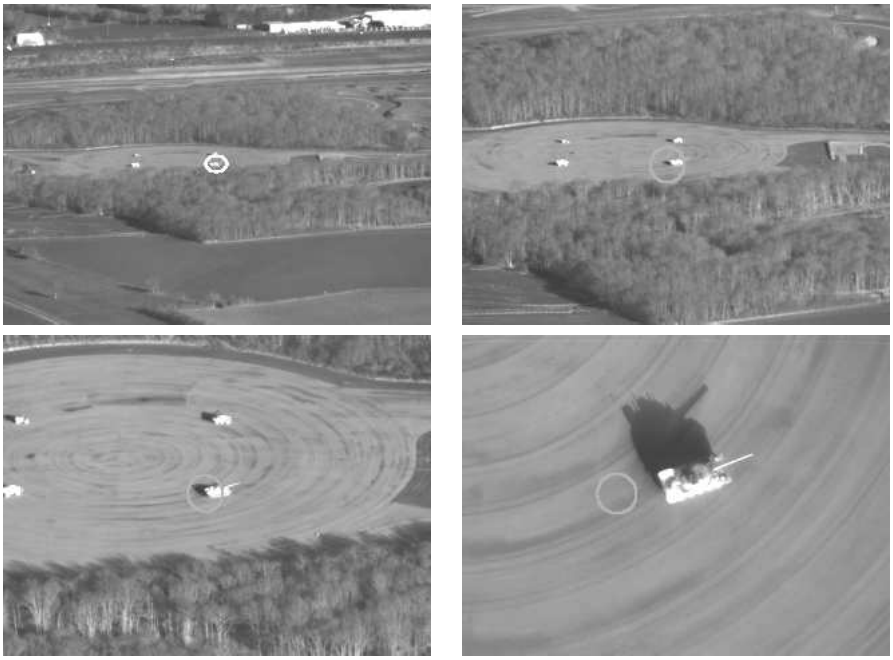

FIG. 5.9 – Meilleures estimées (images 0, 300, 600 et 900).

5.5 Conclusions et discussion

Sur des séquences acquises avec une caméra « standard », les résultats présentés précédemment mettent en évidence que notre système est capable de suivre :

- un objet soumis à d'importants changements d'apparence (des changements d'aspect ou d'orientation) ;
- un objet dans une scène soumise à des changements d'échelle et d'illumination
- un objet se déplaçant à des vitesses variées comme illustré par la séquence avec le joueur de football
- un objet déformable (le joueur de football par exemple)

Les résultats montrent que notre algorithme est robuste pour des applications variées.

Nous avons également testé notre algorithme sur des séquences aériennes (voir le paragraphe 5.4). Sur ces séquences, il apparaît que les difficultés sont différentes que pour le cas classique d'une caméra domestique. Les changements de luminosité sont beaucoup plus importants, les changements d'échelle aussi, et l'algorithme que nous avons proposé ne permet pas toujours de suivre les objets tout au long des séquences. Sur des séquences de synthèse où l'environnement est « contrôlé », le suivi se passe bien ; mais sur les séquences réelles, l'algorithme ne parvient pas à suivre les objets. Une normalisation des histogrammes à l'intensité permettrait peut être de faire face à cette difficulté.

L'algorithme proposé utilise un critère qui permet de détecter le nombre optimal de classes pour les histogrammes dans le but d'effectuer un suivi robuste de divers objets. D'autre part, nous avons mis en place des critères permettant à l'algorithme de fonctionner automatiquement

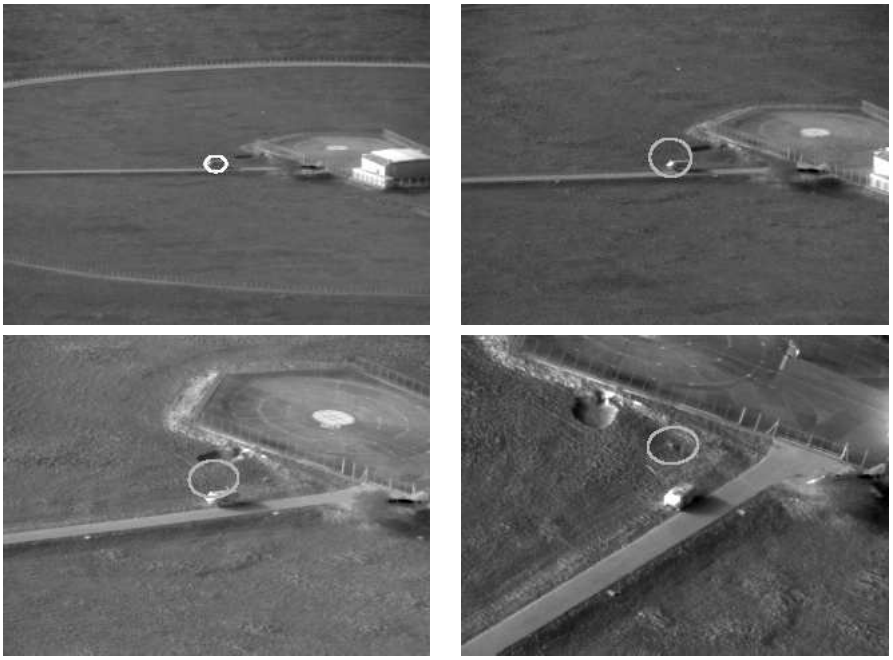

FIG. 5.10 – Meilleures estimées (images 0, 100, 200 et 300).

quelle que soit l'application. Notre approche est une étape vers un suivi entièrement automatique et adaptatif.

Remarquons que nous avons divisé l'ellipse en quartiers ; il serait intéressant de mettre en place un critère similaire à celui utilisé pour sélectionner le nombre de classes de l'histogramme, dans le but de trouver le meilleur compromis entre la quantité d'information spatiale dans le modèle et la flexibilité des histogrammes. D'autre part, il serait utile de pouvoir mettre à jour le nombre de classes au cours de la séquence si nécéssaire (par exemple, si l'objet grossit dans l'image, on pourrait mettre à jour le modèle avec plus de classes pour prendre en compte l'information supplémentaire). Reste à signaler que la mise à jour du modèle en utilisant les distributions de couleurs lors de l'état courant peut poser problème, notamment dans le cas où un objet d'apparence similaire se trouve dans le voisinage.

Enfin, on pourrait facilement étendre l'approche au suivi de plusieurs objets. Si les objets sont de mêmes types (et présentent les mêmes histogrammes), notre approche convient parfaitement : les particules se répartiront naturellement sur les différents objets ; dans le cas où les histogrammes des objets sont différents, il suffit d'initialiser différents objets et de lancer l'algorithme pour chacun des objets.

Chapitre 6

Expérimentations et Validation de l'algorithme de suivi à partir de primitives géométriques

Sommaire

6.1 Introduction

L'objectif de ce chapitre est de présenter un protocole d'évaluation pour notre algorithme de suivi d'objets. Pour cela, il est nécessaire de définir un ensemble de séquences de tests, et une ou plusieurs métriques destinées à quantifier les performances de l'algorithme. Les séquences de tests doivent être accompagnées d'une vérité terrain pour faciliter l'évaluation. Cependant, la caractérisation du type des séquences tests est une tâche difficile puisque les séquences doivent permettre à la fois une évaluation quantitative des performances et aussi des limites de l'algorithme dans un certain cadre applicatif (dans notre cas, le suivi de structures fixes ou d'objets à partir d'images aériennes). La vérité terrain quant à elle permet d'avoir une évaluation quantitative du résultat du suivi, et de comparer éventuellement ces performances à celles obtenues

avec d'autres algorithmes.

Il n'existe pas de critères universels pour évaluer un algorithme de suivi. Nous proposons donc dans ce document un ensemble de critères qui permettent de caractériser les performances du suivi en termes de qualité, robustesse, stabilité et complexité. Notons que ces critères pourraient éventuellement servir à comparer notre algorithme à d'autres algorithmes existants.

6.2 La problématique

En général, la sortie d'un algorithme de suivi est la *localisation* estimée de l'objet à suivre (que l'on appellera par la suite la cible) dans le plan image. Cette localisation est représentée la plupart du temps par une position 2D (coordonnées dans le plan image) mais peut l'être aussi par une forme (2D ou 3D, qui peut être définie par une approximation polygonale, une boite rectangulaire, une ellipse ou autre ...). Dans ce cas, on peut déduire de cette forme une position 2D, en calculant le centre de masse de l'objet ou l'intersection des diagonales de la forme englobante de l'objet par exemple.

On définit par *trajectoire* l'ensemble des positions de la cible ordonnées temporellement. Dans le cas du suivi d'une ellipse par exemple, la trajectoire est constituée de l'ensemble des positions 2D de l'ellipse dans les images ainsi que des paramètres décrivant les ellipses (les deux axes et l'orientation).

Pour mesurer la qualité de la trajectoire obtenue à l'issu du suivi, il est utile de la comparer à une trajectoire de référence. Une vérité terrain est la représentation idéale de la cible au cours de la séquence. Elle est généralement construite à la main. Bien qu'il y ait un certain degré de subjectivité dans la constuction d'une telle vérité terrain (deux personnes peuvent construire une vérité terrain différente pour une même séquence), on considère que la vérité terrain est la représentation idéale de la cible au cours du temps.

6.3 L'évaluation algorithmique : la théorie

L'évaluation algorithmique a pour but de tester un certain nombre de facteurs qui mesurent la qualité des résultats obtenus en sortie d'un algorithme donné. L'objectif est donc d'évaluer les performances de cet algorithme lorsque l'on modifie des paramètres et/ou les données d'entrée. Cette évaluation utilise un certain nombre de métriques qui quantifient la qualité, la robustesse, la stabilité et la complexité de l'algorithme.

La question qui se pose est donc la suivante : comment caractériser la qualité, la robustesse ou la stabilité d'un algorithme et comment les quantifier ?

6.3.1 La qualité

On mesure la *qualité* de l'algorithme en termes d'erreur de position et de taille (dans le cas d'une forme).

L'erreur de position

L'erreur de position, que l'on appele err_{pos} est l'écart entre la trajectoire estimée et la vraie trajectoire (issue de la vérité terrain). En général, cette erreur de position est la distance euclidienne entre les centres de masses estimées et réelles de la cible.

L'erreur de taille

L'erreur de taille, que l'on appelle err_{taille} peut être exprimée de différentes façons : de manière simple on peut mesurer l'écart entre les tailles estimées et réelles de la cible à chaque image et prendre en compte l'erreur d'orientation, notée err_{ori}. Si on veut faire une évaluation plus complète de la taille, on peut compter les pixels détectés comme étant de la cible alors qu'ils ne le sont pas, et inversement les pixels qui n'ont pas été détectés comme étant de la cible et qui le sont. Ces pixels sont appelés respectivement fausses alarmes positives et négatives, et sont notées f_p et f_n. L'erreur de taille est alors normalisée par la taille réelle et prédite de la cible et peut, par exemple, être calculée selon la formule suivante :

$$err_{taille} = \frac{f_p + f_n}{a_r + a_p}$$

où a_r et a_p sont les nombres de pixels de la cible réelle et de la cible prédite.

La qualité, représentée par ces trois mesures $(err_{pos}, err_{taille}, err_{ori})$, est mesurée pour chaque image de la séquence test. Une évaluation visuelle est bien sûr un ajout supplémentaire et il faut la prendre en compte. Les changements éventuels observés au cours de la séquence dans cette mesure de qualité quantifie la *stabilité* de l'algorithme de suivi.

6.3.2 La robustesse

On mesure la *robustesse* de l'algorithme en quantifiant les changements dans les résultats du suivi en fonction des changements des paramètres d'entrée. Ces changements dans les paramètres d'entrée peuvent être des changements de niveau de bruit, des erreurs introduites à l'initialisation, ou d'autres paramètres de l'algorithme.

La robustesse au bruit

La robustesse au bruit, notée rob_{bruit} est testée en ajoutant progressivement du bruit dans les séquences. On peut faire cette évaluation directement sur plusieurs séquences ou par simulation en rajoutant du bruit ou en changeant l'illumination d'une même séquence. On prend pour mesure de la robustesse le ratio entre le nombre d'initialisations et le nombre de bonnes estimations sur la cible en sortie de l'algorithme (dans le sens où l'algorithme a réussi à suivre la cible jusqu'au bout, avec une erreur en position inférieure à un seuil fixé par avance). rob_{bruit} prend des valeurs entre 0 et 1 et plus la valeur est grande, plus l'algorithme est robuste au bruit. L'analyse de cette robustesse permet donc de quantifier la quantité de bruit que l'algorithme peut accepter avant de perdre la cible.

La robustesse à l'initialisation

La robustesse à l'erreur d'initialisation, notée rob_{init} est testée de la manière suivante : en générant un grand nombre d'initialisations aux environs de la cible, on compte le nombre de bons

suivis (comme cela a été défini au paragraphe précédent). De la même façon que précédemment, on peut également ajouter des quantités de bruit variables pour tester la capacité de l'algorithme à retrouver la cible. Les valeurs de rob_{init} sont comprises entre 0 et 1, tout comme la robustesse au bruit.

6.3.3 Le temps d'exécution

Cette mesure sert à quantifier les améliorations apportées à un algorithme ou à comparer plusieurs algorithmes sur une même séquence. Elle est également utilisée pour déterminer le compromis à faire entre le temps d'exécution et la qualité du suivi.

6.4 Expérimentations

Nous avons testé l'algorithme sur des séquences de synthèse et des séquences réelles. L'ensemble des séquences est fournie par Thalès Optronique. D'autre part, nous avons testé les performances de l'algorithme de suivi avec les modèles géométriques, et de celui de l'algorithme intégrant à la fois les modèles géométriques et les histogrammes de niveaux de gris. Nous verrons que la prise en compte des niveaux de gris permet d'améliorer les performances de l'algorithme et le rend efficace pour suivre des objets dans des configurations difficiles, notamment dans le cadre d'images aériennes.

6.5 Avec l'algorithme basé sur des modèles géométriques

Sur les séquences de synthèse, les performances de l'algorithme sont bonnes : la segmentation ne pose pas de probèles, les objets sont bien constrasté par rapport au reste de la scène.

Par contre, sur des séquences réelles, des problèmes de segmentation apparaissent. On constate que dès que la densité de contours à proximité de l'objet est forte, l'algorithme dérive et le suivi échoue. En revanche, si l'objet d'intérêt n'est pas très bien segmenté mais qu'il n'y a pas trop de contours à promiximté, l'alorithme fonctionne bien. Nous allons présenter ces résultats dans les paragraphes suivants.

6.5.1 Sur des séquences de synthèse

Nous avons utilisé deux séquences de synthèse fournies par Thalès Optronique, à basse et moyenne altitude. Nous vous invitons à vous reporter au paragraphe 5.4.1 (page 86) pour la description de ces séquences.

A basse altitude

La figure 6.1 présente les résultats du suivi. Nous avon effectué le suivi avec 120 particules, un bruit en position de 3 pixels sur l'image (le modèle explore l'espace tel que le déplacement sur l'image vaille 3 pixels), et un bruit en taille de 1%. On constate que visuellement, le suivi est bon. Les graphes représentés au bas de la figure 6.1 évaluent la qualité de l'algorithme en terme de position et de taille, ainsi que la répartition des particules. Il y a peu d'erreur en position et en taille et on voit que tout au long de la séquence c'est le modèle du cylindre qui prédomine. Ce résultat est mis en évidence sur le graphe au bas de de la figure 6.1.

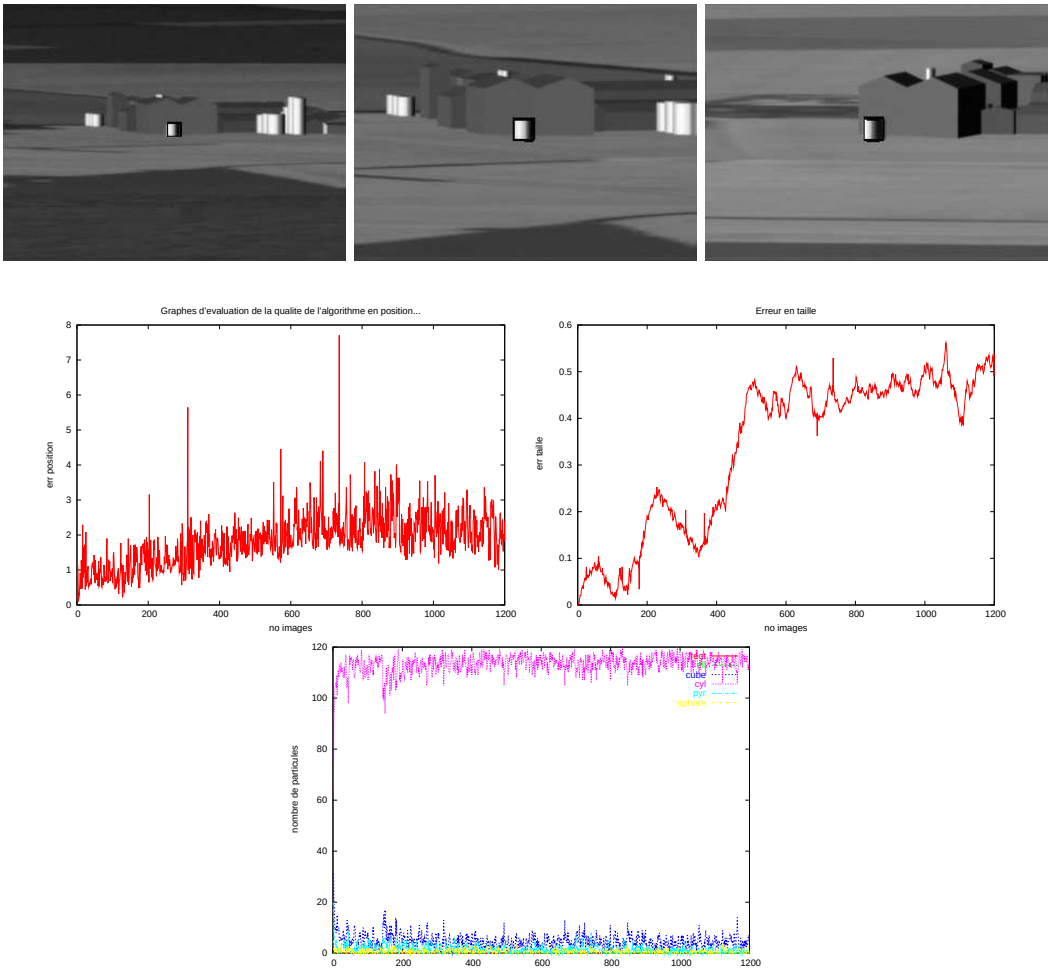

FIG. 6.1 – A basse altitude, résultats du suivi : meilleures estimées sur les images 1, 500 et 1000.

A moyenne altitude

Les objets dans la scène sont donc plus petits, et il est plus difficile de les suivre de manière efficace. Nous suivons deux objets différents : le cylindre (comme précédemment), qui est un objet 3D, et un morceau du sol pour mettre en évidence que l'algorithme est capable de choisir un modèle 2D si cela s'avère nécéssaire.

Les résultats, présentés dans les figures 6.2 et 6.3, montrent que :

- Le suivi du cylindre est difficile à cause de sa petite taille. D'autre part, il est difficile de trouver une manière optimale pour attribuer les poids aux particules. Dans toute la séquence, la meilleure estimée de l'état de l'objet d'intérêt est une pyramide. Cependant, nous avons également représenté sur la figure 6.2 la meilleure particule du bon type (cylindre), et le résultat visuel est satisfaisant.
- Le suivi du morceau du sol est effectué correctement. Le graphe présenté sur la figure 6.3 montre qu'il n'y a aucun doute sur le type de modèle.

FIG. 6.2 – A moyenne altitude, résultats du suivi : meilleures estimées (colonne de gauche) et zoom sur les estimées (colonne de droite) sur les images 1, 500 et 1000.

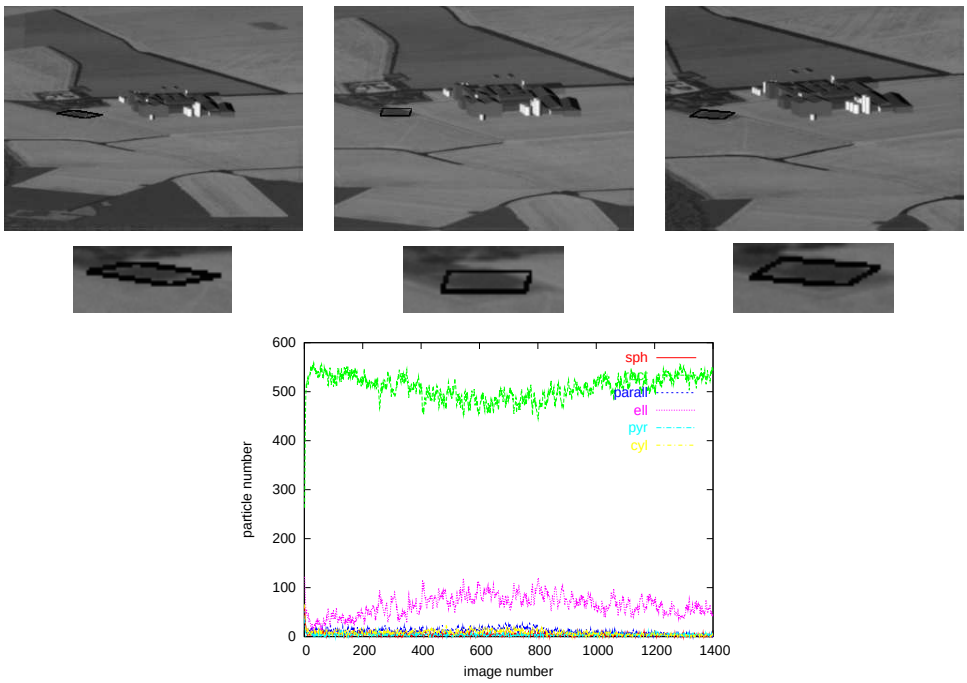

FIG. 6.3 – A moyenne altitude, résultats du suivi : meilleures estimées (colonne de gauche) et zoom sur les estimées (colonne de droite) sur les images 1, 500 et 1000.

6.5.2 Sur des séquences réelles

Description des séquences

Nous avons utilisé deux séquences réelles, fournies par Thalès Optronique. On peut se reporter aux paragraphes 5.4.2 (page 87) et 5.4.2 (page 88) pour leur description.

Les difficultés rencontrées

La principale difficulté est celle de la segmentation des images : dès que la densité de contours est trop grande à proximité de l'objet que l'on veut suivre, cela pose de nombreux problèmes. En

effet, dans ce cas, les modèles vont coller aux contours environnants, et arrivent à se caler bien mieux que sur les contours du modèle que l'on cherche à suivre.

D'autre part, dans une des séquences utilisées, le véhicule que l'on souhaite suivre est placé à coté d'une route. La difficulté est double :

1. les contours du véhicule sont difficiles à extraire, alors que ceux de la route sont très bien extraits.
2. la largeur de la route correspond à la largeur du véhicule

Les modèles ont donc tendance à se caler sur la route et à dévier progressivement. La figure 6.4 montre la segmentation obtenue pour une des premières images de la séquence.

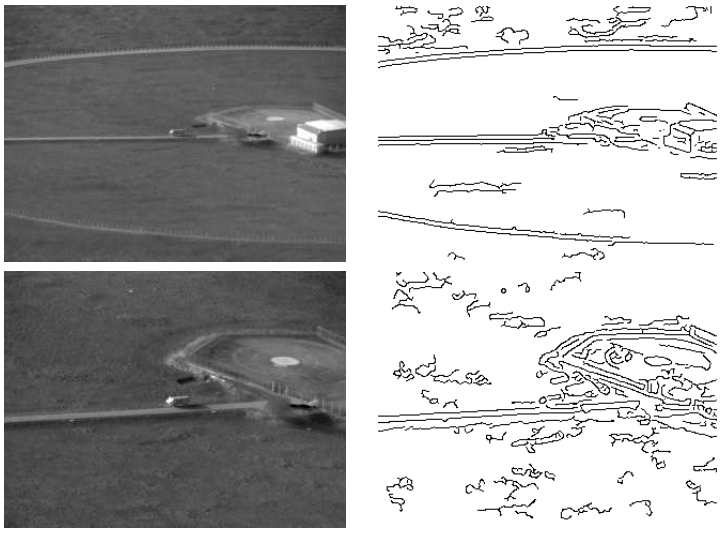

FIG. 6.4 – Le problème de segmentation du véhicule (images 10 et 150)

La dernière difficulté est la présence d'ombres aux environs des véhicules. La segmentation est plus difficile, et les modèles se calent à l'ombre extraite. La figure 6.5 illustre ce problème sur les deux séquences réelles que nous avons utilisé.

Les résultats du suivi

Monter que ca ne marche pas bien !

6.5.3 Conclusions

Sur des séquences de synthèse, où il n'y a pas de problèmes de segmentation et de problèmes d'occultations, l'algorithme fonctionne bien, même à des altitudes élevées. L'algorithme proposé est capable de sélectionner un modèle et de changer de modèle si davantage d'information est disponible, comme dans la séquence à basse altitude.

Par contre, sur des séquences réelles, les problèmes de segmentation et/ou la présence d'ombres dans les images rendent le problème du suivi plus complexe. Les résultats obtenus sur les séquences réelles fournies par Thalès mettent en évidence ces problèmes liés à la segmentation des

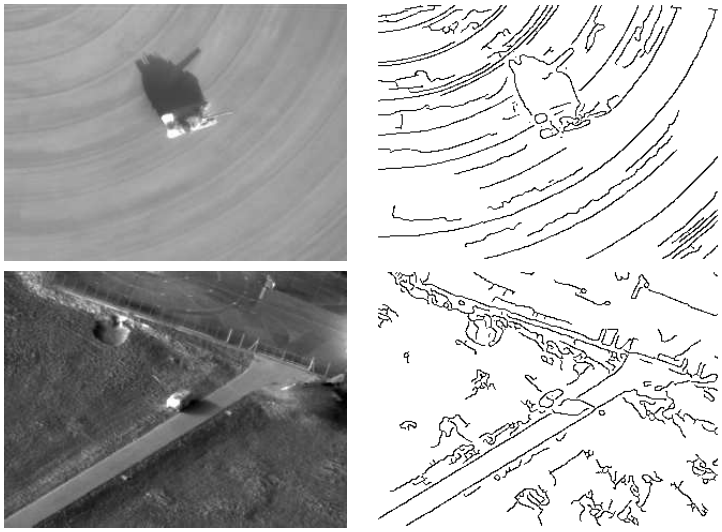

FIG. 6.5 – La présence d'une ombre à coté du véhicule

images. D'autre part, il arrive que la densité de contours extraits soit importante, et les modèles ont tendance à se caler sur des « petits contours » et rétrécissent. Notre algorithme basé sur des modèles géométriques n'est pas suffisamment robuste pour faire face à ces problèmes et à suivre les objets efficacement sur ces séquences réelles. C'est pourquoi nous avons décidé d'intégrer des informations de couleur. L'évaluation de cet algorithme final est donné au prochain paragraphe.

6.6 Avec l'algorithme final

L'algorithme basé uniquement sur les modèles géométriques est très performant lorsque la segmentation se passe bien, c'est-à-dire dans le cas où l'objet d'intérêt est bien segmenté et qu'il n'y a pas de problèmes d'ombres ou d'occultations. Cette configuration « idéale » n'est pas représentative de ce qui se passe en réalité. En effet, dans des séquences aériennes réelles, les difficultés énoncées aux paragraphes précédents montrent les faiblesses de l'algorithme.

Nous avons donc décidé d'intégrer des informations de ninveaux de gris pour rendre l'algorithme plus robuste et performant en cas de mauvaise segmentation. Ce paragraphe a pour but d'évaluer les performances de cet algorithme.

6.6.1 Mise en place de l'évaluation

Avant de présenter les résultats de suivi obtenus sur différents types de séquences, nous commençons par décrire notre procédure d'évalutation. Etant donné le nombre de paramètres de l'algorithme, nous n'avons pas eu le temps d'évaluer la qualité et la robustesse de l'algorithme par rapport à tous les paramètres (robustesse au bruit en position, taille, orientation, nombre de particules, mise à jour des histogrammes ...). Nous n'avons donc pas pu mêtre en place un schéma aussi précis et complet que celui présenté dans la partie théorique au début du chapitre. Néanmoins, nous avons choisi quelques paramètres, qui nous paraissent être les plus importants, pour effectuer une évalutation précise de l'algorithme.

Puisque l'algorithme est basé sur une approche probabiliste, nous l'avons lancé plusieurs fois pour chaque configuration de paramètres, afin d'observer la répétabilité et la stabilité de l'algorithme.

Nous avons tout d'abord évalué le comportement de l'algorithme par rapport au bruit en position mis sur les particules. Ayant fixé un bruit sur la taille raisonnable (bruit de 1% sur la taille du modèle), nous avons fait varier le bruit en position entre 0 et 5 pixels : le bruit en position est donné ici par rapport aux images, de telle sorte que les particules explorent l'espace dans un disque de coté x pixels. Nous avons évalué les erreurs en position par rapport à une vérité terrain « approximative » (l'hypothèse est que la cible est au centre de l'image), et en taille. Ces résultats sont représentés sous forme de courbe dans les prochains paragraphes.

Nous avons également évalué les performances de l'algorithme par rapport au nombre de particules utilisées. L'évaluation du comportement de l'algorithme par rapport au bruit en position a été effectué pour 120 et 300 particules.

D'autre part, il est important d'évaluer le comportement de l'algorithme lorsque l'initialisation est peu précise, voire mauvaise. Nous avons donc consacré une partie à cela, et expliquerons pour chaque type de séquence les difficultés auxquelles nous avons été confrontés.

Par sourci de clarté du document, nous ne présenterons pas tous les résultats dans le chapitre. Nous n'insèrerons ici que les plus représentatifs ou ceux qui illustrent le discours écrit. Nous vous invitons à vous reporter aux annexes pour l'ensemble des résultats.

Enfin, nous allons présenter les résultats de cette évaluation pour une séquence de synthèse et une séquence réelle.

6.6.2 Sur une séquence de synthèse

Nous avons utilisé la séquence de synthèse à basse altitude fournie par Thalès Optronique, dont la description se trouve au paragraphe 5.4.1 (page 86).

Bruit en position

Comme expliqué ci-dessus, nous avons lancé l'algorithme plusieurs fois en faisant varier le bruit en position mis sur les particules. Les autres paramètres de l'algorithme sont réglés de la façon suivante :

- mise à jour des histogrammes toutes les 10 images
- bruit fixé sur la taille du modèle de 1%
- bruit en orientation libre (le modèle peut prendre toutes les orientations possibles)
- nombre de particules 120 et 300

A partir de l'ensemble des résultats obtenus, nous pouvons faire quelques remarques. Tout d'abord, les résultats sont stables : d'une exécution à l'autre, les résultats obtenus sont satisfaisant, le suivi est bien effectué, et cela quelque soit le bruit en position mis sur les particules. Les graphes de la figure 6.6 et les figures 6.7 et 6.8 illustre les résultats obtenus pour 120 particules et un bruit en position de 2 pixels. Sur les deux graphes présentés, l'axe des abscisses correspond aux numéros des images de la séquence et l'axe des ordonnnées aux erreurs par rapport à la

vérité terrain ; ces erreurs sont exprimées en mètres.

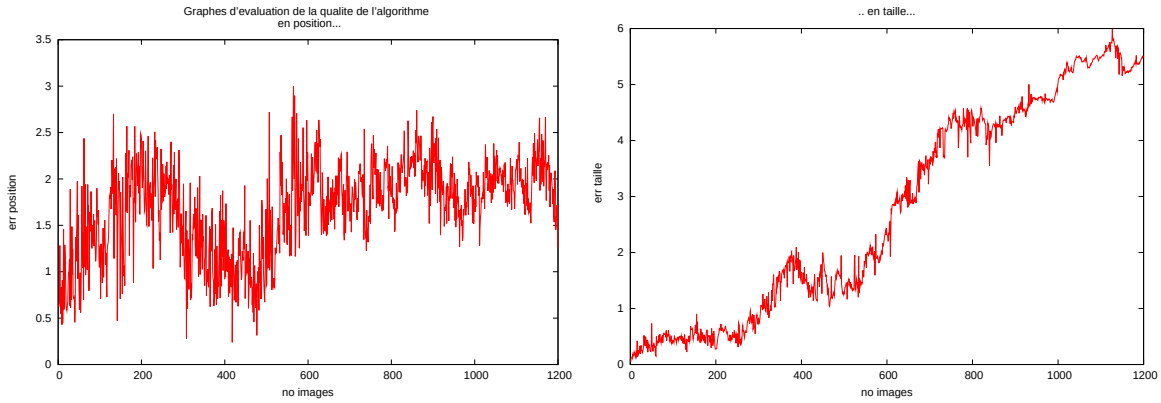

FIG. 6.6 – Graphes avec les parametres suivants : 120 part, bpos=2, btaille=1% .

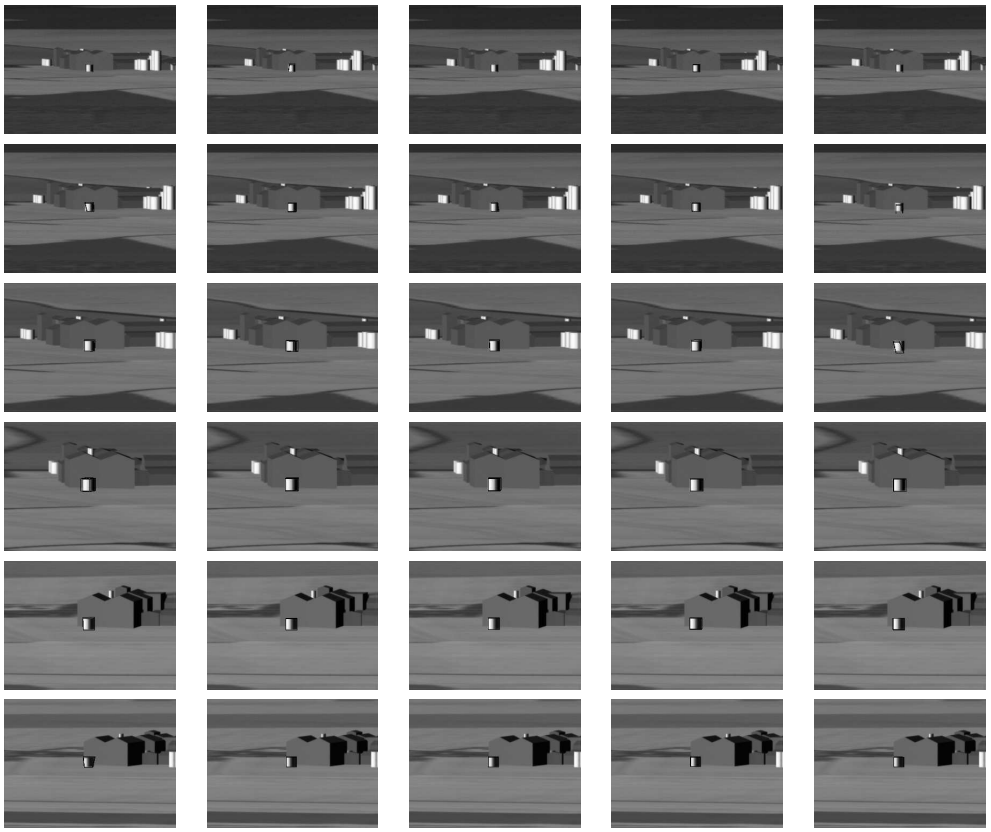

FIG. 6.7 – Meilleures estimées (toutes les 200 images pour nb col exécutions) : 120 part, bpos=2, btaille=1% .

Les deux séries de figures qui suivent représentent, pour chacune des exécutions (une exécution correspond à une colonne d'imagettes) les résultats du suivi obtenus toutes les 100 images. Les

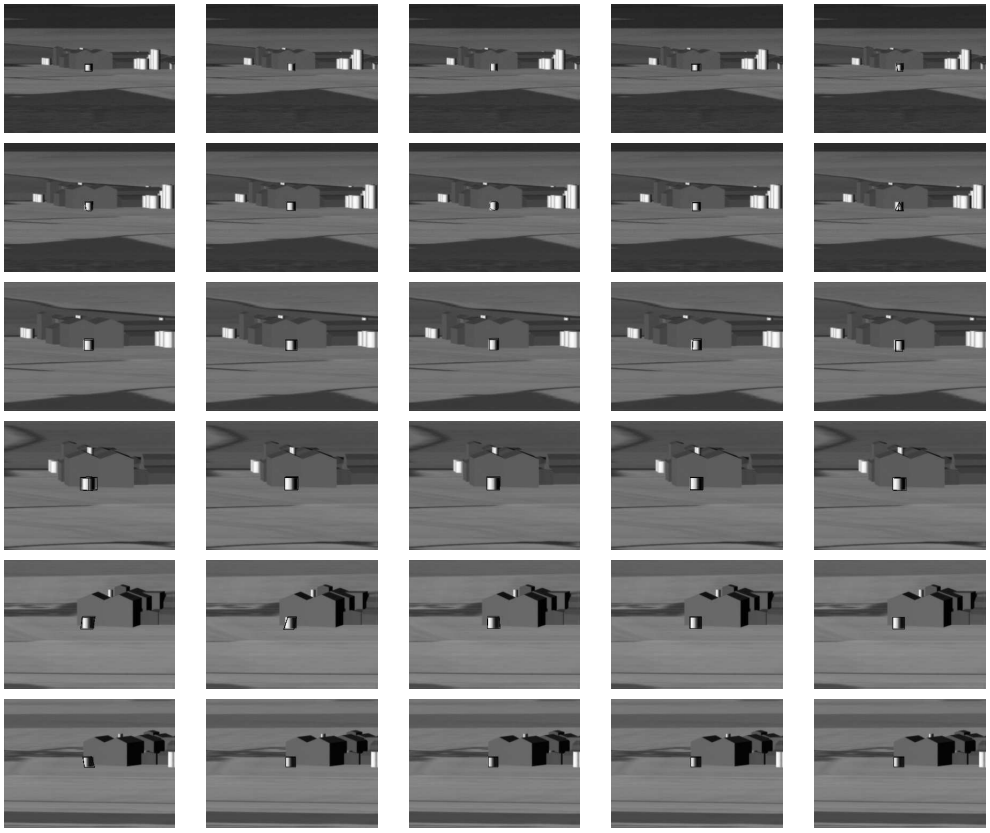

FIG. 6.8 – Estimées moyennes (toutes les 200 images pour nb col exécutions) : 120 part, bpos=2, btaille=1% .

meilleures estimées sont représentées sur la première série (figure 6.7) et les estimées moyennes sur la deuxième série (figure 6.8).

D'autre part, les résultats montrent que la meilleure estimée n'est pas toujours du même type que l'estimée moyenne. Pour le jeu de paramètres 300 particules, bruit en position nul et bruit en taille de 1%, on voit que la meilleure estimée est une pyramide alors que l'estimée moyenne est un cube. Les deux images correspondantes sont représentées sur la figure 6.9, et pour l'ensemble des résultats correspodant à ce jeu de paramètres, nous vous renvoyons aux annexes.

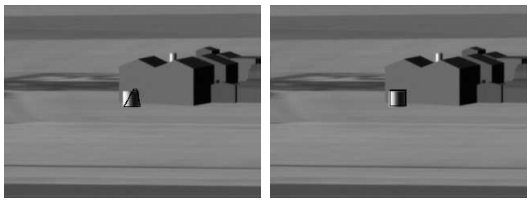

FIG. 6.9 – Meilleure estimée et estimée moyenne pour un même jeu de paramètres (300 part, bpos=0, btaille=1%) : les modèles sont différents .

Evaluation par rapport à l'initialisation

Pour évaluer le comportement de l'algorithme lorsque l'initialisation est peu précise, voire mauvaise, c'est plus compliqué. En effet, si l'initialisation est trop imprécise, l'algorithme utilisant à la fois les informations de contours et celles des niveaux de gris des pixels, l'histogramme calculé dans la première image sera nécéssairement mauvais, et le suivi aura de fortes chance pour ne pas être bon.

Il nous a donc semblé judicieux d'évaluer le comportement de l'algorithme non pas lorsque l'initialisation en position est mauvaise (l'histogramme calculé sera forcément mauvais), mais plutôt lorsque l'initialisation sur la taille est mauvaise ou, du moins, imprécise.

Pour cela, nous avons lancé l'algorithme avec un pourcentage d'erreur sur la taille à l'initialisation. En mettant un bruit suffisamment important sur les particules, on évalue la capacité de l'algorithme à converger vers la taille réelle du modèle. Sur la séquence à basse altitude, l'objet d'intérêt est un cube de 10 mètres de coté. Nous avons lancé l'algorithme avec 20% d'erreur sur la taille (à la fois pour des modèles initiaux plus petits et plus grands), et pour chaque configuration nous avons fait plusieurs exécutions. Pour cette expérimentation, les paramètres sont réglés de la façon suivante :

- mise à jour de l'histogramme toutes les 10 images
- bruit en position de 3 pixels
- bruit en taille de 5%
- 120 particules

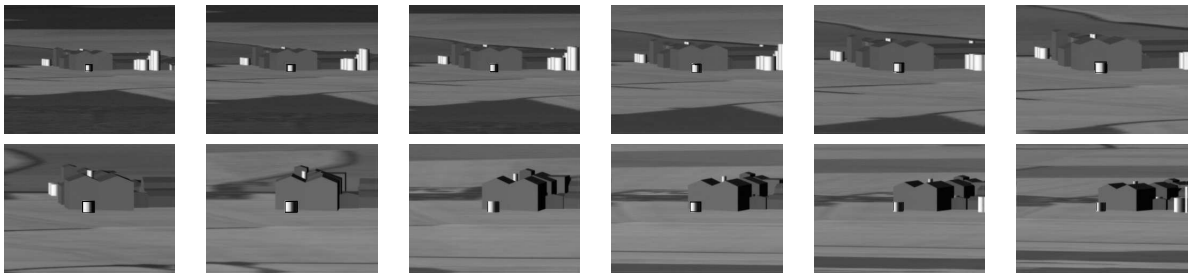

FIG. 6.10 – Suivi à basse altitude pour une initialisation des modèles avec 20% d'erreur sur la taille (plus petit) : le suivi se passe bien

Les résultats obtenus sont satisfaisants : dans l'ensemble, le suivi est bon visuellement. Dans aucun cas, l'algorithme ne perd complètement la cible. En analysant plus précisément les résultats, on peut faire les remarques suivantes : dans le cas où l'initialisation est faite avec un modèle plus petit, l'erreur sur la taille qui est observée (et qui apparait sur les graphes) est en réalité une erreur sur la profondeur estimée du modèle. Dans cette séquence, on voit que l'erreur en largeur et en hauteur est faible ; ceci s'explique par la bonne segmentation des images. Les particules réussisent à évoluer de manière à coller aux contours extraits dans les images. Par contre, pour la profondeur, c'est plus compliqué puisque même visuellement nous n'avons pas beaucoup d'information dessus. Un exemple de suivi est présenté sur la figure 6.10.

De manière générale, lorsque l'initialisation se fait avec un modèle plus grand, l'erreur en

taille est plus faible mais l'erreur en position plus grande. Sur les cinq exécutions que nous avons effectué, trois suivis ont été très bien effectués. Les deux autres suivis sont moins bons, sélectionnant d'avantage de gros modèles et d'un mauvais type (la pyramide) ; ceci dit, à la fin de la séquence la mauvais estimation du départ est réparée, et le suivi se passe correctement ; la fgure 6.11 illustre cette remarque. Cela montre que, progressivement, l'algorithme a réussi à faire le compromis entre se rapprocher des contours extraits dans les images et mettre à jour l'histogramme pour enlever du fond et ne garder que l'information pertinente. Il serait intéressant de voir dans ce cas si l'augmentation du nombre de particules permettrait d'obtenir de meilleurs résultats.

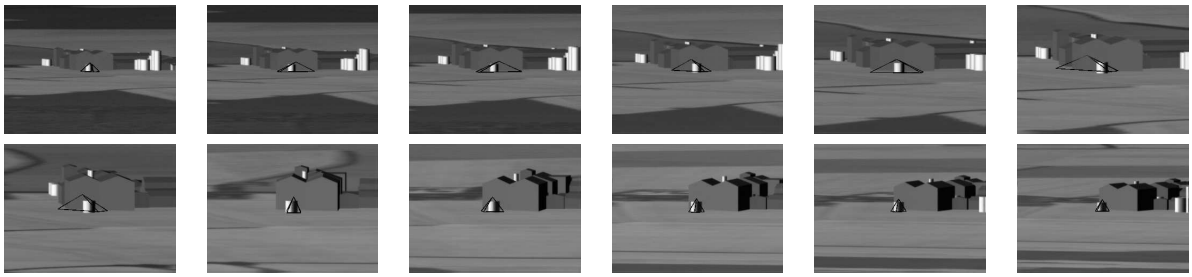

FIG. 6.11 – Suivi à basse altitude pour une initialisation des modèles avec 20% d'erreur sur la taille (plus grand) : même avec une mauvaise initialisation, l'algorithme réussit à se « recaler » et à suivre l'objet

Bruit en taille

Nous avons réalisé le même type d'expériences que pour l'évaluation par rapport au bruit en position réalisée précédemment. Nous avons fixé le nombre de particules à 300, le bruit en position à 3 pixels, et avons fait varier le bruit sur la taille entre 0% et 10%. L'ensemble des résultats est présenté dans les annexes.

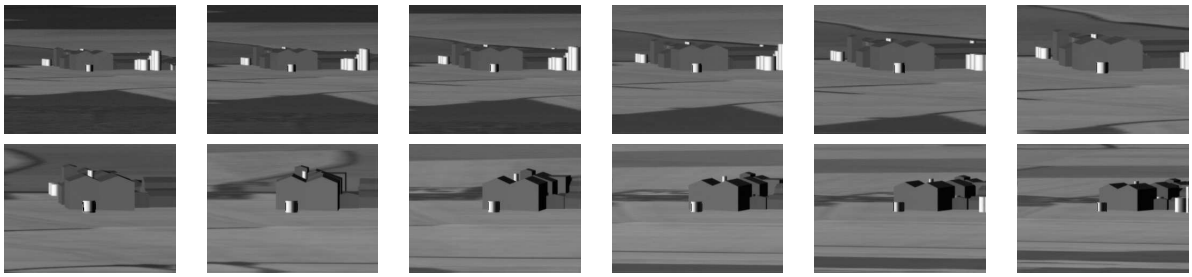

FIG. 6.12 – Meilleures estimées (toutes les 100 images pour nb col exécutions) : 300 part, bpos=3, btaille=9% .

Nous pouvons, à partir des résultats obtenus, faire quelques remarques. Tout d'abord, l'algorithme est robuste par rapport au bruit sur la taille ; en effet, même avec un bruit proche de 10%, l'algorithme continue de bien suivre l'objet d'intérêt. Ceci s'explique bien sûr par le fait que la segmentation des images ne pose pas vraiment de problèmes, et que l'objet d'intérêt est

bien contrasté par rapport au reste de la scène.

La figure 6.12 présente les résultats du suivi obtenu pour un bruit de 9%. Visuellement le suivi est bon ; le graphe correspondant à l'erreur sur la taille, et qui est représenté sur la figure 6.13, montre qu'il existe bien une erreur sur la taille. Cette erreur correspond à l'erreur qui est faite sur la profondeur du modèle. Il n'y a pas assez d'information sur cette séquence pour estimer précisément la taille du modèle en profondeur.

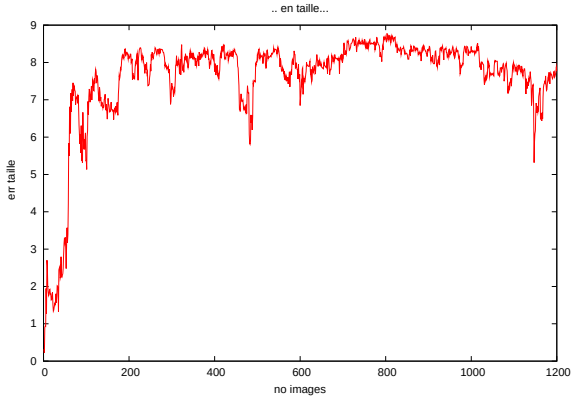

FIG. 6.13 – Graphes représentant l'erreur sur la taille avec les parametres suivants : 300 part, bpos=3, btaille=9% .

Robustesse par rapport au nombre de particules

Pour la majorité des expériences, nous avons lancé l'algorithme pour 120 et 300 particules. Sur cette séquence, les difficultés sont peu nombreuses, et même avec relativement peu de particules, l'algorithme parait robuste au bruit (en position comme en taille) et/ou à une initialisation peu précise.

Cependant, pour des séquences réelles où les problèmes sont plus nombreux et où le problème du suivi devient plus difficile, l'ajout de particules permet de rendre l'algorithme plus robuste à certains paramètres. Cette remarque « intuitive » sera confirmée dans les paragraphes qui suivent, dont l'objectif est d'effectuer la même évaluation mais sur une séquence réelle.

6.6.3 Sur une séquence réelle

Comme nous venons de l'expliquer, l'objectif de ce paragraphe est d'effectuer une évaluation semblable à celle effectuée sur la séquence de sythèse à basse altitude, mais cette fois-ci sur une séquence réelle. La séquence qui a été utilisé pour cela est celle qui avait été précédemment décrite au paragraphe 5.4.2).

Nous évaluons donc le comportement de l'algorithme pour différents bruits en position et en taille mis sur les particules, ainsi que les performance de l'algorithme lorsque l'initialisation et imprécise.

Bruit en position

Comme nous l'avons fait pour la séquence de synthèse, nous avons lancé l'algorithme plusieurs fois en faisant varier le bruit en position mis sur les particules. Les autres paramètres de l'algorithme sont réglés de la façon suivante :

- mise à jour des histogrammes toutes les 10 images
- bruit fixé sur la taille du modèle de 1%
- bruit en orientation libre (le modèle peut prendre toutes les orientations possibles)
- nombre de particules 120 et 300

L'ensemble des résultats sont présentés dans les annexes. Nous ne présenterons ici que les plus représentatifs, ou ceux qui permettent d'illustrer notre discours. Néanmoins, nous vous invitons à regarder les annexes pour plus de détails sur les résultats obtenus.

A partir de l'ensemble des résultats obtenus pour cette série de tests, nous pouvons faire quelques remarques. Tout d'abord, nous constatons que l'ajout de particules rend l'algorithme plus robuste.

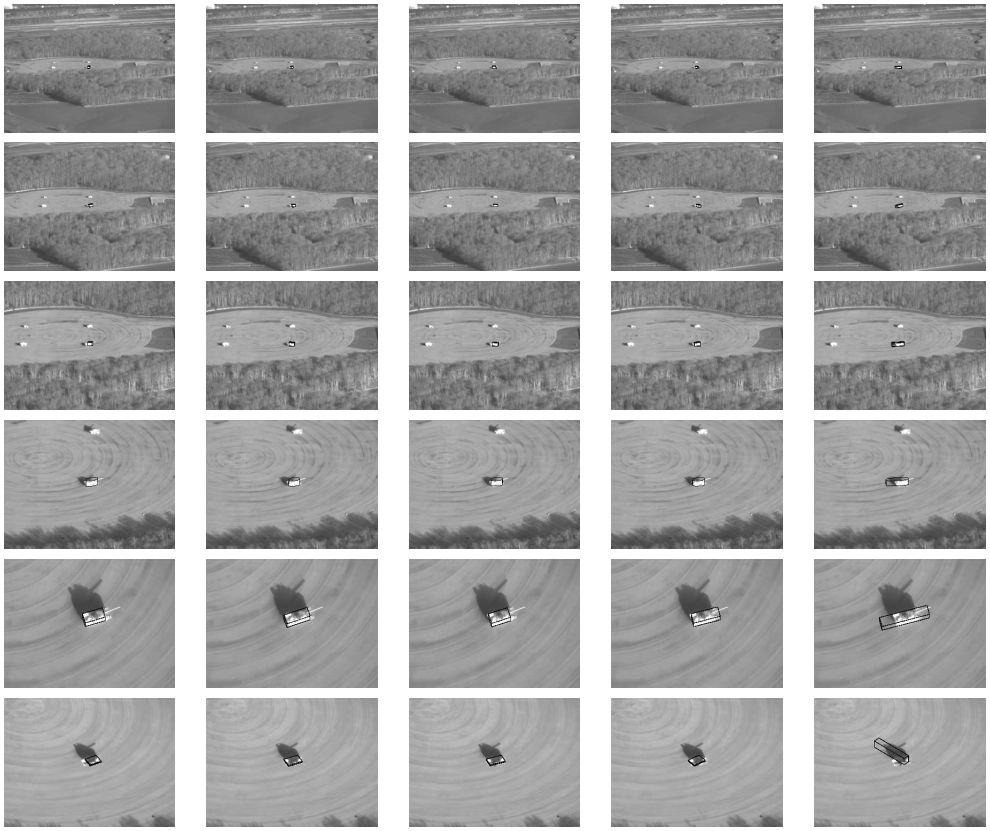

FIG. 6.14 – Meilleures estimées (toutes les 200 images pour nb col exécutions) : 300 part, bpos=3, btaille=1% .

.

D'autre part, et c'est sans doute la conclusion la plus importante de cette expérimentation, les résultats obtenus mettent en évidence la sensibilité de l'algorithme par rapport au bruit en position. D'une exécution à l'autre, les résultats obtenus fluctuent. Il arrive que le suivi se passe très bien pour une exécution, mais pas pour les autres (pour un jeu de paramètres fixé). Néanmoins, il existe une configuration de paramètre pour laquelle le suivi est stable, et se passe bien. Les résultats obtenus pour ce jeu de paramètres sont représentés sur les figures 6.14, 6.16 et 6.15.

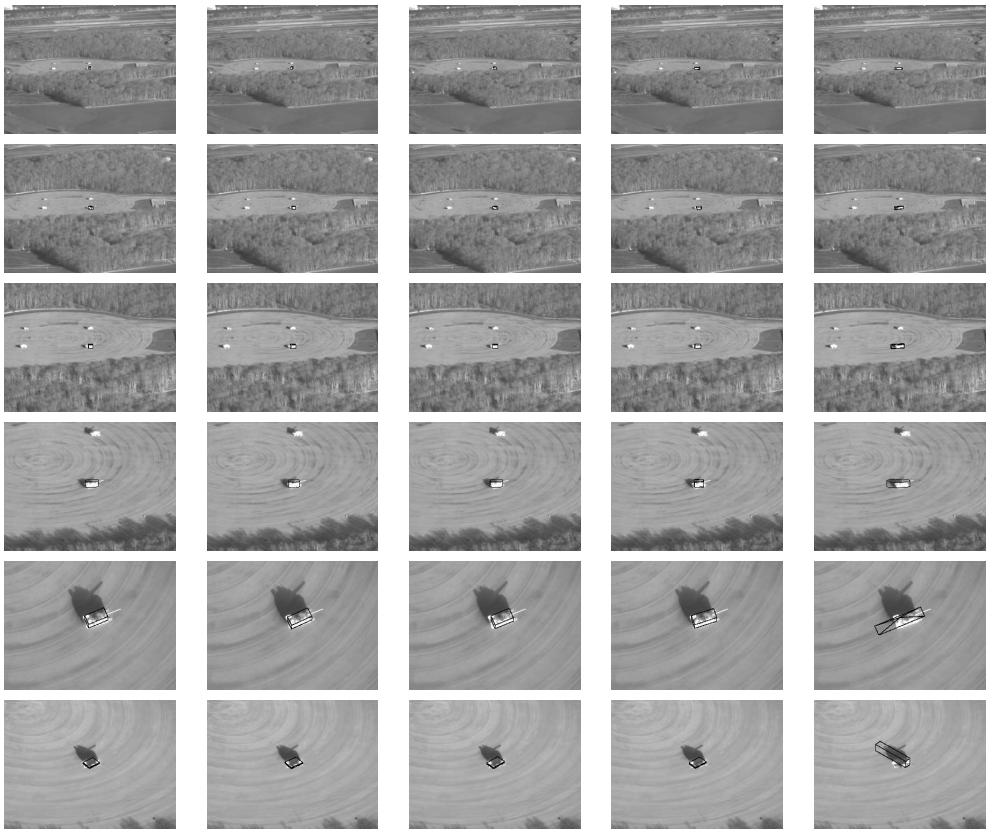

FIG. 6.15 – Estimées moyennes (toutes les 200 images pour nb col exécutions) : 300 part, bpos=3, btaille=1% .
.

En analysant un peu plus précisément les deux graphes de la figure 6.16, on constate que l'erreur en position augmente beaucoup aux alentours de l'image 900. Cela correspond au moment où l'ombre, située à coté du véhicule, se distingue nettement du reste de la scène, et est relativement bien segmentée par rapport au véhicule. Les particules ont tendance à s'accrocher à l'ombre, et leur position dévie un peu.

L'erreur sur la taille provient quant à elle de la mauvaise estimation de la hauteur du véhicule. L'information 3D de la scène n'est pas suffisante ici pour estimer précisément la hauteur du véhicule.

Comme nous l'avions déjà remarqué pour la séquence de synthèse, il arrive que la meilleure particule ne soit pas du même type que celui de l'estimée moyenne ; c'est le cas illustré sur la

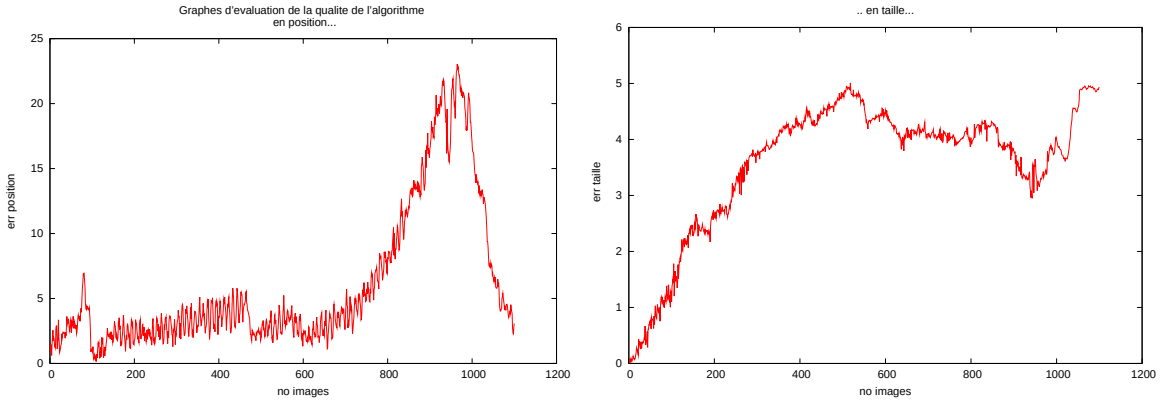

FIG. 6.16 – Graphes avec les parametres suivants : 300 part, bpos=3, btaille=1% .

.

figure 6.17 : on voit que la meilleure estimée est un cube alors que l'estimée moyenne est une pyramide.

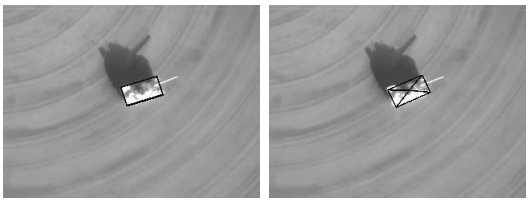

FIG. 6.17 – Meilleure estimée et estimée moyenne pour un même jeu de paramètres (300 part, bpos=4, btaille=1%) : les modèles sont différents

.

Evaluation par rapport à l'initialisation

Nous avons procédé de la même manière que pour la séquence de synthèse, et avons uniquement étudié le comportement de l'algorithme face à une mauvaise initialisation de la taille du modèle. Cependant, la séquence réelle présentant de nombreuses difficultés, nous avons initialisé l'algorithme avec une erreur de 10% sur la taille (alors que nous l'avions fait avec une erreur de 20% dans le cas de la séquence de synthèse).

De même que pour les autres séries de tests, nous avons lancé plusieurs fois l'algorithme pour chaque jeu de paramètres (donc ici pour chaque taille initiale de modèle), en fixant les autres paramètres de l'algorithme avec les valeurs suivantes :

- 300 particules
- bruit en position de 3 pixels
- bruit en orientation libre
- bruit en taille de 5%
- mise à jour de l'algorithme toutes les 10 images

Les résultats obtenus, présentés en annexe, sont satisfaisants. Lorsque le modèle est initialisé avec 10% d'erreur sur la taille (trop petits), dans 80% des cas, le suivi se passe bien, l'algorithme ne perd pas la cible. Visuellement le suivi est bien réalisé, comme on peut le voir sur la figure 6.18.

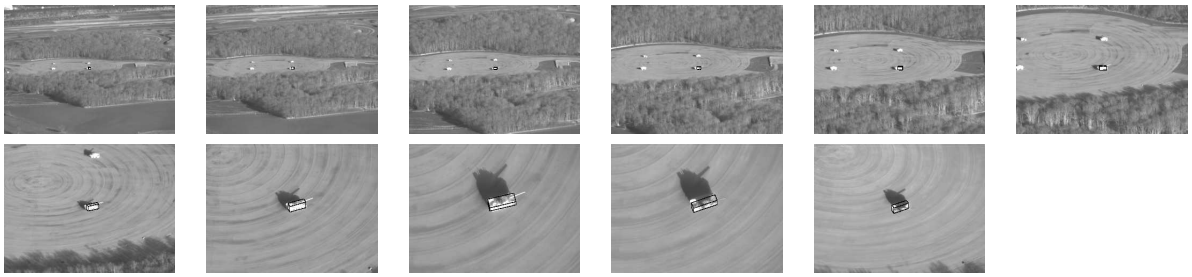

FIG. 6.18 – Résultats du suivi pour une initialisation des modèles avec 10% d'erreur sur la taille (plus petit) : le suivi se passe bien

Lorsque le modèle est initialisé avec des modèles plus grand (toujours avec 10%) d'erreurs, le suivi se passe bien pour 80% des cas. Les graphes sont semblables à ceux des modèles initialisés « trop petits » et les résultats visuels se ressemblent aussi. La figure 6.19 présente le résultat d'un suivi réussi dans ce cas là.

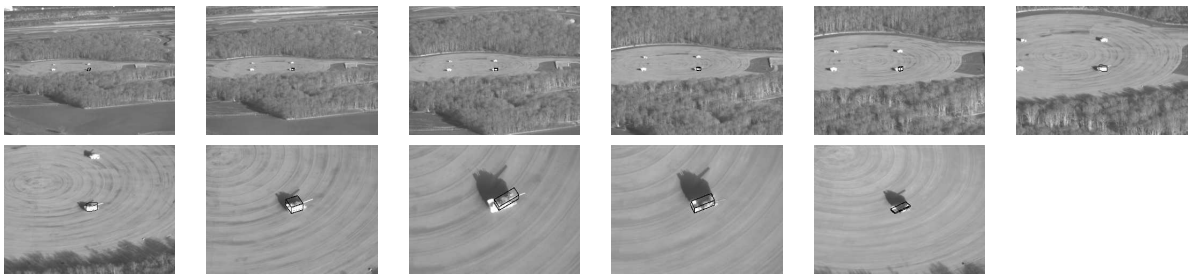

FIG. 6.19 – Résultats du suivi pour une initialisation des modèles avec 10% d'erreur sur la taille (plus grand) : le suivi se passe bien

Bruit en taille

Il reste à évaluer les performances de l'algorithme lorsque l'on rajoute du bruit sur la taille du modèle. Nous avons fait varier le bruit entre 0% et 10%. L'ensemble des résultats est présenté en annexes, et nous allons ici tirer quelques conclusions de cette série de tests.

Commentaires ? ?

Robustesse par rapport au nombre de particules

Dans le cas de séquences réelles comme celle étudiée ici, l'ajout de particules permet de rendre l'algorithme plus robuste au bruit principalement. Nous avons évalué l'algorithme avec 120 et 300 particules. Il serait intéressant de rajouter des particules pour voir si les performances sont

bien meilleures ou pas. C'est encore un compromis à faire, puisque rajouter des particules permet davantage de robustesse mais génère beaucoup plus de calculs .... Nous discuterons de cela dans le chapitre de conclusions.

6.7 Conclusions

L'évaluation que nous avons réalisée est loin d'être complète. Les paramètres de l'algorithme sont nombreux, et il n'est pas possible de faire une évaluation précise par rapport à chacun de ces paramètres, cela prendrait un temps bien trop long. Nous avons donc choisi un certain nombre de critères qu'il nous a paru intéressant d'évaluer : la robustesse au paramètres de bruit (en position ainsi qu'en taille), le nombre de particules et le comportement de l'algorithme face à une mauvaise initialisation.

De cette évaluation nous pouvons tirer un certain nombre de conclusions : sur des séquences où la segmentation ne pose pas de problèmes, l'algorithme est performant et robuste au bruit. C'est ce que nous avons constaté sur la séquence de synthèse, un cas « idéal », mais qui permet de valider notre approche dans des conditions simples.

Sur des séquences réelles, de nouvelles difficultés apparaissent : la segmentation est plus compliquée, et les ombres qui peuvent intervenir rendent le suivi plus difficile. Néanmoins, nous avons trouvé un jeu de paramètres pour lequel le suivi se passe bien, et pour lequel les résultats sont répétables.

Bien sûr, l'algorithme est sensible aux bruits (sur la position et sur la taille) dans le cas où les contours extraits sont nombreux. Pour un jeu de paramètres qui permet le fonctionnement de l'algorithme, nous avons vu que l'algorithme était relativement robuste au bruit sur la taille. Les dernières expérimentations montrent qu'un bruit de 7% n'empêche pas l'algorithme de suivre sa cible (dans la plupart des cas) sur la séquence réelle que nous avons testé.

Conclusions et Perspectives

1 Conclusions

Dans la première partie de ce document, nous avons présenté les défis à relever lorsque l'on veut faire du suivi d'objets à partir de séquences d'images. Les difficultés sont nombreuses, et l'application finale détermine souvent une partie des choix qui sont faits pour la modélisation. Le choix d'un modèle adapté à l'objet que l'on veut suivre est difficile, mais détermine en grande partie les performances du suivi. Les problèmes d'occultations, de changements de luminosité de la scène ou de changement d'échelle sont souvent difficiles à résoudre. Quant à la stratégie suivie pour l'initialisation ou la mise à jour du modèle, la littérature regorge d'exemples divers.

Au lieu de parler de choix, on pourrait plutôt parler de compromis à faire. Le meilleur choix correspond au meilleur compromis. Par exemple, pour la mise à jour du modèle : si on souhaite effectuer un suivi sur une longue séquence, au cours de laquelle les changements de luminosité ou les changements d'échelle sont importants, il est nécéssaire de mettre à jour le modèle. Comme on l'a expliqué dans le manuscrit, si le modèle n'est jamais mis à jour, un changement dans l'orientation du modèle dans la scène (ou quelqu'autre changement d'apparence de l'objet dans la scène) suffit à rendre l'algorithme incapable de continuer à suivre l'objet d'intérêt. Au contraire, si l'on décide de mettre à jour le modèle à chaque image, en supposant que les changements ne sont pas trop importants d'une image à l'autre, on pourrait supposer que le modèle mis à jour serait toujours adéquat ; ce n'est malheureusement pas le cas ; le modèle, s'il est mis à jour trop régulièrement, n'est plus robuste et dérive progressivement avec le temps. Et, en général, le résultat du suivi n'est pas optimal. Y a t-il une solution optimale ? Dépend-elle de l'application visée ? La question de la mise à jour du modèle se pose donc comme un compromis à faire, entre accepter de dériver un peu ou de ne pas être robuste à des effets trop importants (en termes de luminosité ou d'échelle). La mise à jour du modèle est un exemple, mais on trouve des compromis à faire dans le cadre du suivi d'objets à partir d'une vidéo à tous les niveaux : dans le choix du modèle (qui s'il est trop complexe ne pourra pas être utilisé pour d'autres applications, et s'il est trop simple ne sera pas assez discriminant), le choix pour la modélisation du ou des mouvements dans la scène, d'un modèle de luminosité, ...

Nous avons, dans un second chapitre, rappelé les bases et l'historique du filtrage. L'estimation d'un système dynamique à partir d'observations bruitées est un problème complexe. Le filtre bayésien optimal fournit une expression récursive exacte de la loi *a posteriori* de l'état conditionnellement aux données d'observations. Cette expression n'est cependant calculable explicitement que lorsque des hypothèses restrictives sont faites sur les systèmes considérés : pour des systèmes linéaires gaussiens, le filtre de Kalman est la solution au problème du suivi. Nous avons également vu que lorsque certaines hypothèses sont relachées, d'autres filtres permettent de calculer la loi recherchée. Lorsque les systèmes ne sont ni linéaires, ni gaussiens (on se place donc dans

le cas général), l'application de ces filtres aboutit dans la plupart des cas à une divergence du suivi, et il faut trouver d'autres méthodes pour estimer l'état du système. Les méthodes statistiques fournissent dans ce cas des outils permettant le calcul d'une approximation de cette loi ; leur principe repose sur la représentation de la loi par un ensemble d'échantillons (appelées particules) situés en des positions de l'espace d'état. Une première étape fait évoluer le nuage des particules dans l'espace des états, et une deuxième permet la sélection des particules pertinentes. Différents filtres ont été proposés, et diffèrent par leur méthode de sélection de particules.

Nous avons ensuite présenté nos contributions. Nous avons proposé trois algorithmes de suivi d'objets basés sur un filtrage particulaire. Le premier algorithme utilise des histogrammes de niveaux de gris. Le second utilise des modèles géométriques, et une matrice de transition permet à l'algorithme de changer de modèle lorsqu'il l'estime nécéssaire. Enfin, le dernier algorithme est une combinaison des deux algorithmes précédents. Les informations de niveaux de gris et de contours sont intégrées à un filtrage particulaire pour faire du suivi robuste d'objets. Les algorithmes ont été évalués d'une part sur des séquences « de la vie courante » accessibles sur internet, et d'autre part, sur des séquences de synthèse et des séquences réelles fournies par Thalès Optronique. Ces dernières séquences ont permis de valider les performances de nos algorithmes dans le cadre du suivi d'objets à partir d'images aériennes, dans un contexte bien spécifique. En effet, les difficultés avec des séquences aériennes sont différentes, et résident dans les changements d'apparence, de luminosité et d'échelle qui sont beaucoup plus importants qu'à l'ordinaire. Lorsque l'avion est loin, l'objet d'intérêt n'est représenté que par quelques pixels dans l'image, mais au fur et à mesure qu'il se rapproche, sa taille augmente jusqu'à représenter le tiers ou la moitié de l'image.

En conclusion, l'algorithme basé sur les histogrammes est performant pour des séquences dans lesquelles l'objet d'intérêt est soummis à des changements d'échelle, d'illumination ou d'apparence, se déplaçant à des vitesses variées, lorsque la caméra est une caméra standard. Sur des séquenes d'images aériennes où l'objet n'est représenté que par quelques pixels au début, le suivi se passe bien, mais l'ellipse (représentant l'objet) a tendance à grossir et à intégrer du fond dans le modèle.

L'algorithme du suivi basé sur les modèles géométrique est performant quand les contours de l'objet d'intérêt sont bien extraits ou que la densité de contours à proximité de l'objet n'est pas trop grande. Le fait de pouvoir changer de modèle si nécéssaire présente un grand intérêt, notamment sur les séquences d'images aériennes pour lesquelles des changements d'échelle sont très importants. L'avantage de se baser sur les contours plutôt que sur l'intensité des pixels est que l'algorithme est plus robuste aux changements de luminosité que les algorithmes basés sur la couleur (même si la segmentation dépend aussi beaucoup des intensités des pixels, un changement d'illumiation a moins d'effet sur des approches basées contours que des approches basées couleur).

Enfin, l'algorithme combinant les informations de couleur et de contour permet de rendre le suivi plus robuste en cas de mauvaise segmentation. **... A compléter**

Notons que les séquences qui étaient à notre disposition ne présentaient pas de cas d'occultations. A ce sujet, on peut quand même dire que si l'occultation est partielle, l'algorithme devrait être capable de suivre l'objet (le modèle géométrique rétrécirait) ; si l'ocultation est totale, *a priori* l'algorithme ne devrait pas pouvoir suivre l'objet.

2 Perspectives

Les algorithmes proposés dans ce document sont loin d'être parfaits, et s'il nous restait du temps, nous pourrions effectuer quelques améliorations. D'autre part, nos algorithmes permettent d'éffectuer du suivi robuste d'objets à partir d'une caméra, que les images soient des images aériennes ou prises par une caméra standard que l'on trouve dans le commerce. Cependant, ils pourraient être utilisés pour d'autres applications : c'est ce que nous allons présenter dans ces deux sous sections.

2.1 Améliorations

En ce qui concerne les améliorations que l'on pourrait apporter à nos différents algorithmes, un élément important est la prise en compte des changements d'illumination dans la scène. Le terme « changements d'illumination » utilisé ici comprend à la fois les changements de luminosité de la scène (un nuage qui passe devant la caméra par exemple), et des changements d'illumination apparent, dus à un changement de point de vue. Ainsi, pour les deux algorithmes utilisant les histogrammes de niveaux de gris, une solution pourrait être de rendre ces histogrammes robustes aux changements d'illumination en trouvant la meilleure transformation de luminosité pour que les histogrammes (de la particule à l'image courante et du modèle) soient les plus proches au sens de la distane de Bhattacharyya utilisée pour comparer nos histogrammes.

Dans l'algorithme utilisant les histogrammes de niveaux de gris, de la même manière qu'on a trouvé un critère permettant de déterminer automatiquement le nombre de classes adapté pour les histogrammes, on pourrait trouver un critère pour trouver la meilleure partition de l'ellipse (la division de l'ellipse en quartiers n'est surement pas la partition optimale dans la plupart des cas).

Dans l'algorithme utilisant les modèles géométriques, nous avons fixé les paramètres de la matrice de transition à la main. Une autre façon de faire serait de procéder par apprentissage pour fixer automatiquement les paramètres de cette matrice de transition. **(réfléchir à des pistes ! !)**

Dans notre algorithme intégrant à la fois les niveaux de gris et les contours, les deux informations sont prises en compte au même niveau et une note est associée à chacune de ces informations. On pourrait procéder d'une toute autre manière, et faire un suivi en utilisant les histogrammes, puis là où les particules sont « bonnes », envoyer des particules des différents modèles géométriques. Le suivi par histogramme permet d'avoir une bonne localisation des objets, et les modèles géométriques permettent ensuite d'obtenir des informations 3D sur la taille du modèle et sa localisation. D'autre part, en ce qui concerne la mise à jour de l'histogramme dans cet algorithme, elle est actuellement effectuée toutes les N_h images ; ce serait bien de trouver un critère pour pouvoir le mettre à jour de manière automatique. Un dernier point encore sur cet algorithme concerne le problème de l'initialisation : on pourrait effectuer au début un suivi purement 2D en utilisant les histogrammes, puis trouver un critère traduisant qu'il faut passer à la 3D parce qu'assez d'information est disponible. On pourrait alors générer des modèles dans l'ellipse (sous l'hypothèse vérifiée en pratique que l'ellipse contient l'objet d'intérêt).

2.2 Autres applications

Nous avons présenté trois algorithmes de suivi d'objets. Pour cela, nous avons utilisé des méthodes statistiques, qui permettent de modéliser les données avec une grande flexibilité. Les

algorithmes mis au point pourraient être utilisés pour d'autres applications que celles du suivi d'objets à partir d'une caméra.

Par exemple, on pourrait se servir de ces algorithmes pour corriger l'estimation de la trajectoire de l'avion. La centrale inertielle de l'avion fournit des informations, et des estimations sont faites sur les positions de l'avion dans la scène, ainsi que de celle de la cible. Ces estimations sont très souvent erronnées. Et pour les corriger, on pourrait procéder de la manière suivante : supposons que l'on connaisse avec précision le modèle de l'objet à suivre, il suffit ensuite de lancer des particules en mettant du bruit non pas sur les paramètres de taille de l'objet, mais sur les paramètres d'orientation du modèle et sur sa position. Une (ou plusieurs) particule bien placée permet de remonter à une correction de l'estimation initiale.

Une autre application a été soulevée lors d'une présentation effectuée chez Thalès avec des responsables de différents projets. L'application visée est celle de la reconnaissance d'objets lors d'une mission effectuée par un drone. Dans ce cas, on imagine que l'on charge dans le drone juste avant la mission le modèle de l'objet à retrouver dans la scène, et on envoie des particules de ce modèle dans l'espace vu par le drone. Ou encore, on lui donne plusieurs modèles pour voir ce qu'il peut reconnaître dans la scène. Ce sont des applications qui pourraient utiliser une ou plusieurs parties de nos travaux, et qu'il faudrait préalablement adapter un peu pour qu'elles fonctionnent.

Annexe A

Calcul des matrices de projections à partir des fichiers de CPDV

L'objectif de ce document est de présenter le calcul des matrices de projection à partir des fichiers de CPDV (Conditions de Prises De Vues) de TOSA.

A.1 La problématique

Supposons que l'on ait une caméra à une position connue $\mathbf{t}$ pointant sur un point $\mathbf{Q}$. On suppose d'autre part que le repère monde est attaché au plan du sol avec l'axe des Z orienté vers le haut. L'objectif est de calculer la matrice de rotation de la caméra à partir de ces informations, pour remonter ensuite à la matrice de projection de la caméra.

On assume d'autre part qu'il n'y a pas de rotation autour de l'axe optique. Cela équivaut à dire que la ligne verticale (virtuelle) passant par $\mathbf{Q}$ est verticale dans l'image.

Enfin, on fait l'hypothèse que la caméra regarde vers le bas, donc que $\mathbf{Q}$ est plus bas que $\mathbf{t}$; autrement dit, $\mathbf{Q}_3 \leq \mathbf{ts}_3$.

A.2 La solution !

La matrice de projection s'écrit sous la forme :

$$\mathsf{P} \sim \mathsf{R}(\mathsf{I} - \mathbf{t}) \tag{A.1}$$

Première contrainte La première contrainte qu'on utilise est que la ligne verticale passant par le point $\mathbf{Q}$ est verticale aussi dans l'image. Cela revient à dire que le point de fuite dans la direction verticale est « au dessus » du point principal, et à une coordonnées en x nulle.

D'où :

$$\left(\mathsf{P} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right)_1 = 0$$

En développant cette expression, on en tire directement :

$$\mathsf{R}_{13} = 0$$

On considère d'autre part que la caméra regarde « vers le bas », ce qui veut dire que la deuxième coordonnée de la direction verticale dans l'image est négative (autrement dit, l'axe des y dans l'image est dirigé vers le bas, par conséquent un point au dessus du point principal à une coordonnée en y négative).

Deuxième contrainte Notre deuxième contrainte est la suivante : $\mathbf{Q}$ est projeté sur le point principal (rappelons que le point principal est l'origine dans le plan image) :

$$\mathsf{P}\begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \\ \mathbf{Q}_3 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

En développant, on obtient :

$$\mathsf{R}(\underbrace{\mathbf{Q} - \mathbf{t}}_{\mathbf{D}}) \sim \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{A.2}$$

Posons $\mathbf{E}$ le vecteur de norme 1 tel que : $\mathbf{E} = \lambda\mathbf{D}$. Il existe deux tels vecteurs (égaux au signe près), choisissons celui pour lequel $\lambda > 0$.

L'ambiguïté du signe dans l'équation (A.2) est maintenant remplacée par une ambiguïté sur le signe : $\mathsf{R}\mathbf{E} = \pm....$ Or on a :

$$\mathsf{R}^{\mathsf{T}}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \pm\mathbf{E}$$

Nous pouvons donc écrire la matrice de rotation R de la façon suivante :

$$\mathsf{R} = \begin{pmatrix} 1 & & \\ & 1 & \\ & & \pm 1 \end{pmatrix}\begin{pmatrix} \mathsf{R}_{11} & \mathsf{R}_{12} & 0 \\ \mathsf{R}_{21} & \mathsf{R}_{22} & \mathsf{R}_{23} \\ \mathbf{E}_1 & \mathbf{E}_2 & \mathbf{E}_3 \end{pmatrix}$$

Contraintes d'orthonormalité R étant une matrice de rotation, ses première et troisième lignes sont orthogonales. Cela nous donne la relation :

$$\mathsf{R}_{11}\mathbf{E}_1 + \mathsf{R}_{12}\mathbf{E}_2 = 0$$

ce qui peut encore s'écrire :

$$\begin{pmatrix} \mathsf{R}_{11} \\ \mathsf{R}_{12} \end{pmatrix} = \mu\begin{pmatrix} \mathbf{E}_2 \\ -\mathbf{E}_1 \end{pmatrix}$$

pour un μ donné.

D'autre part, la première ligne de $\mathbf{R}$ doit être de norme 1 donc :

$$\mu^2\left(\mathbf{E}_1^2 + \mathbf{E}_2^2\right) = 1$$

En tenant compte du fait que $\mathbf{E}$ aussi est de norme 1, on obtient :

$$\mu = \pm \frac{1}{\sqrt{1 - \mathbf{E}_3^2}}$$

La matrice R a donc la forme suivante :

$$\mathsf{R} = \begin{pmatrix} \pm 1 & & \\ & 1 & \\ & & \pm 1 \end{pmatrix} \begin{pmatrix} \frac{\mathbf{E}_2}{\sqrt{1-\mathbf{E}_3^2}} & -\frac{\mathbf{E}_1}{\sqrt{1-\mathbf{E}_3^2}} & 0 \\ \mathsf{R}_{21} & \mathsf{R}_{22} & \mathsf{R}_{23} \\ \mathbf{E}_1 & \mathbf{E}_2 & \mathbf{E}_3 \end{pmatrix}$$

Il reste donc à calculer la deuxième ligne de R. On l'obtient (au signe près) par le produit vectoriel des deux autres lignes. Ainsi,

$$\begin{pmatrix} \mathsf{R}_{21} \\ \mathsf{R}_{22} \\ \mathsf{R}_{23} \end{pmatrix} = \pm \frac{1}{\sqrt{1 - \mathbf{E}_3^2}} \begin{pmatrix} \mathbf{E}_2 \\ -\mathbf{E}_1 \\ 0 \end{pmatrix} \times \mathbf{E} = \pm \frac{1}{\sqrt{1 - \mathbf{E}_3^2}} \begin{pmatrix} -\mathbf{E}_1\mathbf{E}_3 \\ -\mathbf{E}_2\mathbf{E}_3 \\ 1 - \mathbf{E}_3^2 \end{pmatrix}$$

On obtient finalement pour la matrice de rotation R :

$$\mathsf{R} = \begin{pmatrix} \pm 1 & & \\ & \pm 1 & \\ & & \pm 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{1-\mathbf{E}_3^2}} & & \\ & \frac{1}{\sqrt{1-\mathbf{E}_3^2}} & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_2 & -\mathbf{E}_1 & 0 \\ -\mathbf{E}_1\mathbf{E}_3 & -\mathbf{E}_2 E_3 & 1 - \mathbf{E}_3^2 \\ \mathbf{E}_1 & \mathbf{E}_2 & \mathbf{E}_3 \end{pmatrix} \qquad \text{(A.3)}$$

Cela nous donne donc 8 possibilités pour la matrice R (pour les différentes combinaisons $\pm$).

Trouver une solution unique Tout d'abord, R étant une matrice de rotation, son déterminant doit être égal à 1. Dans l'équation précédente, le déterminant de la matrice droite est toujours négatif ; par conséquent, la matrice de gauche doit avoir elle aussi un déterminant négatif, ce qui élimine 4 des 8 solutions possibles. Il ne nous reste que 4 possibilités :

$$\begin{pmatrix} -1 & & \\ & -1 & \\ & & -1 \end{pmatrix} \begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix}$$

Nous allons maintenant utiliser une des hypothèses énoncées au début : la caméra regarde vers le bas, ce qui signifie que la deuxième coordonnée (du point de fuite) est négative. (Rappelons que l'axe des y dans l'image pointe vers le bas ; par conséquent, un point au dessus du point principal à une coordonnée en y négative). Cela se traduit par :

$$\frac{\left(\mathsf{P}\begin{pmatrix}0\\0\\1\\0\end{pmatrix}\right)_2}{\left(\mathsf{P}\begin{pmatrix}0\\0\\1\\0\end{pmatrix}\right)_3} < 0$$

On a :

$$\mathsf{P}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} \sim \begin{pmatrix}\pm 1 & & \\ & \pm 1 & \\ & & \pm 1\end{pmatrix}\begin{pmatrix}0\\ \sqrt{1-\mathbf{E}_3^2}\\ \mathbf{E}_3\end{pmatrix}$$

Etant donné notre hypothèse $\mathbf{Q}_3 < \mathbf{t}_3$, nous avons $\mathbf{E}_3 < 0$. Par conséquent, les deux derniers coefficients de la matrice diagonale de gauche doivent être du même signe pour satisfaire la contrainte ; cela laisse donc deux solutions pour la matrice diagonale, qui sont :

$$\begin{pmatrix}-1 & & \\ & -1 & \\ & & -1\end{pmatrix}\begin{pmatrix}-1 & & \\ & 1 & \\ & & 1\end{pmatrix}$$

Pour choisir entre ces deux solutions il faut encore utiliser une hypothèse : le point $\mathbf{Q}$ est devant la caméra. En traduisant cela en coordonnées 3D homogènes, cela donne :

$$
\begin{aligned}
\begin{pmatrix} \mathsf{R} & -\mathsf{R}\mathbf{t} \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \\ \mathbf{Q}_3 \\ 1 \end{pmatrix} &= \begin{pmatrix} \mathsf{R}\,(\mathbf{Q}-\mathbf{t}) \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} \mathsf{R}\mathsf{R} \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} \frac{1}{\lambda}\mathsf{R}\mathbf{E} \\ 1 \end{pmatrix} \\
&= \frac{1}{\lambda} \begin{pmatrix} -\frac{1}{\sqrt{1-\mathbf{E}_3^2}} & & \\ & \pm\frac{1}{\sqrt{1-\mathbf{E}_3^2}} & \\ & & \pm 1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_2 & -\mathbf{E}_1 & 0 \\ -\mathbf{E}_1\mathbf{E}_3 & -\mathbf{E}_2\mathbf{E}_3 & 1-\mathbf{E}_3^2 \\ \mathbf{E}_1 & \mathbf{E}_2 & \mathbf{E}_3 \end{pmatrix} \begin{pmatrix} \mathbf{E}_1 \\ \mathbf{E}_2 \\ \mathbf{E}_3 \end{pmatrix} \\
&= \frac{1}{\lambda} \begin{pmatrix} -\frac{1}{\sqrt{1-\mathbf{E}_3^2}} & & \\ & \pm\frac{1}{\sqrt{1-\mathbf{E}_3^2}} & \\ & & \pm 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ 0 \\ \frac{\pm 1}{\lambda} \\ 1 \end{pmatrix}
\end{aligned}
$$

Le point est devant la caméra si la troisième coordonnée (la coordonnée en z sur l'axe optique de la caméra qui pointe vers la scène) est positive. Lorsque $\lambda > 0$, le signe $+$ doit être choisi. Finalement, la seule matrice possible est :

$$
\begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix}
$$

L'unique solution pour la matrice de rotation R est donc :

$$
\mathsf{R} = \begin{pmatrix} \frac{-1}{\sqrt{1-\mathbf{E}_3^2}} & & \\ & \frac{1}{\sqrt{1-\mathbf{E}_3^2}} & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_2 & -\mathbf{E}_1 & 0 \\ -\mathbf{E}_1\mathbf{E}_3 & -\mathbf{E}_2\mathbf{E}_3 & 1-\mathbf{E}_3^2 \\ \mathbf{E}_1 & \mathbf{E}_2 & \mathbf{E}_3 \end{pmatrix} \tag{A.4}
$$

A.3 En résumé

Pour calculer la matrice de projection à partir des CPDV, il faut donc :

1. Calculer $\mathbf{D} = \mathbf{Q} - \mathbf{t}$.

2. Calculer λ :

$$\lambda = \frac{1}{\sqrt{\mathbf{D}\mathbf{D}^T}}$$

3. Calculer $\mathbf{E} = \lambda\mathbf{D}$.
4. Calculer R selon l'équation (A.4).
5. Calculer P selon l'équation (A.1).

Annexe B

Le passage de la 2D à la 3D dans notre approche

B.1 Problème

Notre base de donnée de modèles contient des modèles 2D (rectangles et ellipses), et des modèles 3D (parallélépipèdes, cylindres, pyramides, sphères). Une matrice de transition permet de passer d'un modèle à un autre. Le problème est le suivant : puisque chaque modèle possède des paramètres qui lui sont propres, quels paramètres faut-il modifier, et comment faut-il les modifier pour que le modèle du nouveau type soit cohérent avec le modèle de l'ancien type ?

Pour le passage d'un modèle 2D à un autre modèle 2D, cette opération est simple ; pour le passage d'un modèle 3D à un autre modèle 3D aussi. Le problème se pose donc lorsque l'on passe d'un modèle 2D à un modèle 3D, ou que l'on passe d'un modèle 3D à un modèle 2D.

B.2 Première solution

La première solution qui fut implémentée est simpliste et peu (voire pas du tout) réaliste. Supposons la configuration suivante : on passe d'un modèle 2D à un modèle 3D. La solution la plus simple est de tirer aléatoirement une hauteur et de générer un modèle 3D avec la hauteur tirée, les autres paramètres restant en accord avec ceux du modèle 2D initial. La figure suivante permet de représenter cette première solution implémentée.

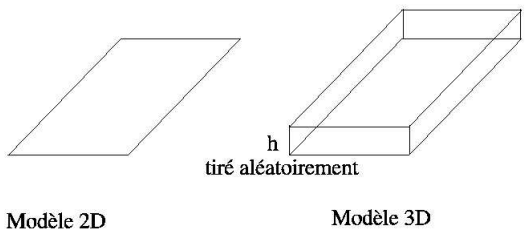

FIG. B.1 – Illustration de la première solution

Cette stratégie fonctionne mal : la probabilité que la hauteur tirée aléatoirement convienne

est très faible ; le modèle 3D généré n'a donc que peu de chances de survivre. L'amélioration qui a été faite consiste à générer à partir du modèle 2D un grand nombre de modèles 3D avec des hauteurs tirées aléatoirement, et de ne garder que le meilleur modèle 3D (au sens du meilleur score, comme défini au paragraphe 4.4.3).

B.3 Solution plus réaliste

B.3.1 L'idée

La solution précédente est peu réaliste puisque les contours du modèle 3D généré n'ont aucune raison de correspondre dans l'image avec les contours du modèle 2D reprojeté dans l'image, ni même avec les contours de l'image. Pour pallier à ce problème, on décide d'utiliser le cône de vue de la caméra pour faire la transition entre le modèle 2D initial et le modèle 3D que l'on veut générer. Le principe est le suivant : il faut que les contours du modèle 2D reprojetés dans l'image correspondent aux contours du modèle 3D reprojetés dans cette image. Par conséquent on utilise le cône de vue de la caméra pour modifier les paramètres de longueur et largeur initiaux ainsi que la position du centre du nouveau modèle de façon à faire correspondre les contours des modèles dans l'image. Ceci est illustré sur la figure ci-dessous.

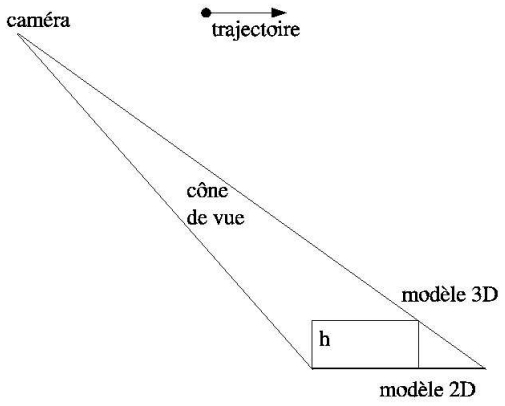

FIG. B.2 – Illustration de la solution réaliste : coupe selon un des axes principaux du modèle

B.3.2 L'implémentation

Rappelons brièvement les différentes étapes de notre algorithme de filtrage particulaire. L'initialisation étant supposée faite, on procède pour chaque nouvelle image de la manière suivante :

- Pour chaque particule i :
  1. On détermine, grâce à la matrice de transition, son nouveau type de modèle
  2. On modifie ses paramètres pour qu'ils soient cohérents avec l'ancien modèle
  3. On la propage grâce à la dynamique de mouvement
  4. On évalue la confiance que l'on a dans cette particule

– On procède à l'étape de rééchantillonnage de l'ensemble des particules

L'implémentation décrite dans ce paragraphe correspond donc à l'étape numéro 2 sur le schéma précédent. Lorsque l'on détecte le passage d'un modèle 2D à un modèle 3D, il faut modifier un ensemble de paramètres pour que la transition entre les deux modèles soit cohérente. Pour cela, on commence par déterminer le point le plus éloigné de la caméra ; notons ce point **A**. On cherche à trouver un point **B** tel que les conditions suivantes soient satisfaites :

– Le point **B** doit appartenir au nouveau modèle désiré (modèle 3D), et doit être situé au dessus du plan support (à une hauteur h non nulle)
– Les points **A** et **B** doivent se projeter dans l'image au même point

La figure suivante illustre ces deux contraintes.

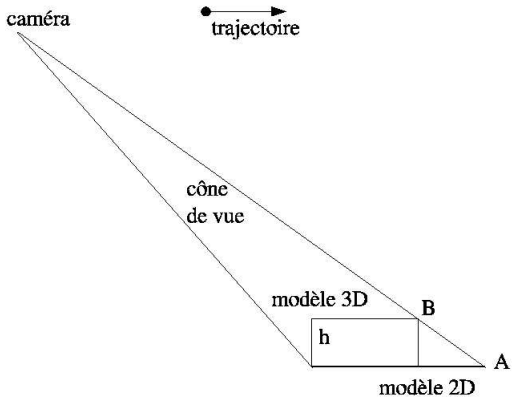

FIG. B.3 – Contraintes sur le point **B**

On tire une hauteur h aléatoire. Puis on utilise les deux contraintes en même temps ; on sait que le point doit être situé à une hauteur h du plan, h non nulle. On en tire donc $\mathbf{B}_3 = h$.

D'autre part, on sait que le point doit se projeter dans l'image au même endroit, qu'il appartienne au modèle 2D ou au modèle 3D. Il existe donc un facteur α tel que :

$$\mathsf{P}\mathbf{A} = \alpha \mathsf{P}\mathbf{B}$$

ce qui s'écrit encore

$$\mathsf{KR}(\mathsf{I}|-\mathbf{t}) \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ 1 \end{pmatrix} = \alpha \mathsf{KR}(\mathsf{I}|-\mathbf{t}) \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ 1 \end{pmatrix}$$

En simplifiant un peu, on obtient :

$$(I|-\mathbf{t}) \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ 1 \end{pmatrix} = \alpha (I|-\mathbf{t}) \begin{pmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ 1 \end{pmatrix}$$

Puis, en développant, on obtient le système d'équations suivant :

$$\begin{cases} \mathbf{A}_1 - \mathbf{t}_1 &= \alpha(\mathbf{B}_1 - \mathbf{t}_1) \\ \mathbf{A}_2 - \mathbf{t}_2 &= \alpha(\mathbf{B}_2 - \mathbf{t}_2) \\ \mathbf{A}_3 - \mathbf{t}_3 &= \alpha(\mathbf{B}_3 - \mathbf{t}_3) \end{cases}$$

Puisque $\mathbf{A}_3$ et $\mathbf{B}_3$ sont connus (ils valent respectivement 0 et h si on suppose que le point $\mathbf{A}$ est sur le plan $z = 0$, sinon ils valent $\mathbf{A}_3$ et $\mathbf{A}_3 + h$), on peut en déduire α. On a donc

$$\alpha = \frac{\mathbf{A}_3 - \mathbf{t}_3}{\mathbf{B}_3 - \mathbf{t}_3} = \frac{\mathbf{A}_3 - \mathbf{t}_3}{\mathbf{A}_3 + h - \mathbf{t}_3}$$

α connu, il est alors facile de remonter aux coordonnées du point $\mathbf{B}$.

$$\begin{cases} \mathbf{B}_1 &= \dfrac{(\mathbf{A}_1 + (\alpha - 1)\mathbf{t}_1)}{\alpha} \\ \mathbf{B}_2 &= \dfrac{(\mathbf{A}_2 + (\alpha - 1)\mathbf{t}_2)}{\alpha} \\ \mathbf{B}_3 &= h \end{cases}$$

Il reste à calculer les nouvelles dimensions du modèle, et la position du centre du modèle. Intuitivement, puisque le modèle « prend de la hauteur », ses dimensions sur les deux axes de base (x et y) vont diminuer. On suppose que l'on garde l'orientation du modèle ; en notant lx_{init}, ly_{init} les dimensions du modèle 2D et lx_{new}, ly_{new}, et h_{new} les dimensions du nouveau modèle 3D, on a les relations suivantes :

$$\begin{aligned} lx_{new} &= lx_{init} - 0.5\sqrt{(\mathbf{A}_1 - \mathbf{B}_1)^2} \\ ly_{new} &= ly_{init} - 0.5\sqrt{(\mathbf{A}_2 - \mathbf{B}_2)^2} \\ h_{new} &= h \end{aligned}$$

De la même manière, puisque les dimensions du modèle 3D sur les deux axes de base sont plus petites, la position du centre du nouveau modèle sera plus proche de la caméra que ne l'était l'ancien. On a donc

$$\begin{cases} x_{new} &= x_{init} + 0.5\sqrt{(\mathbf{A}_1 - \mathbf{B}_1)^2} \\ y_{new} &= y_{init} + 0.5\sqrt{(\mathbf{A}_2 - \mathbf{B}_2)^2} \\ z_{new} &= 0 \end{cases}$$

Remarques Nous avons utilisé cette méthode pour passer de l'un des deux modèles 2D à un modèle 3D de même forme (parallélépipède ou cylindre). L'implémentation pour les modèles de type pyramide et sphères consiste simplement à tirer des hauteurs h aléatoirement et à ne garder que la meilleur particule (au sens du meilleur poids calculé comme cela a été défini au paragraphe 4.4.3).

B.4 Passage de la 3D à la 2D

En procédant de manière similaire, on peut transformer un modèle 3D en modèle 2D tel que les contours de chacun des deux modèles se reprojetent dans l'image aux mêmes endroits. Cette fois-ci, en gardant les notations précédentes, on part d'un point $\mathbf{B}$ connu (déterminé comme étant le point du modèle 3D à la hauteur h du modèle le plus éloigné de la caméra), et on cherche à déterminer les coordonnées du point $\mathbf{A}$ appartenant au plan du sol qui se projette dans l'image

au même endroit que le point **B**.

En utilisant les équations précédentes, on détermine les coordonnées du point **A**. Les nouvelles dimensions du modèle sont données par les relations suivantes :

$$\begin{array}{rcc} lx_{new} & = & lx_{init} + 0.5\sqrt{(\mathbf{A}_1 - \mathbf{B}_1)^2} \\ ly_{new} & = & ly_{init} + 0.5\sqrt{(\mathbf{A}_2 - \mathbf{B}_2)^2} \\ h_{new} & = & 0 \end{array}$$

Et la nouvelle position du centre du modèle 2D par :

$$\left\{ \begin{array}{rcc} x_{new} & = & x_{init} - 0.5\sqrt{(\mathbf{A}_1 - \mathbf{B}_1)^2} \\ y_{new} & = & y_{init} - 0.5\sqrt{(\mathbf{A}_2 - \mathbf{B}_2)^2} \\ z_{new} & = & 0 \end{array} \right.$$

Bibliographie

[Aka73] H. Akaike. Information theory as an extension of the maximum likelihood principle. *Second International Sysposium on Information Theory*, pages 267–281, 1973.

[AM79] B.D.O. Anderson and J.B. Moore. *Optimal filtering.* Englewood Cliffs, NJ : Prentice-Hall, 1979.

[Ati84] K. Atika. Image sequence analysis of real world human motion. *Pattern recognition*, 17(1) :73–83, 1984.

[BC04] T. Bréhard and J.P. Le Cadre. Un problème de filtrage non linéaire avec variance d'état inconnue : le pistage mono-cible par mesure d'angle seul. *XXXVIèmes Journées de Statistique de la SFDS*, 2004.

[BD95] B. Bascle and R. Deriche. Region tracking through image sequences. In *International Conference on Computer Vision (ICCV95)*, pages 302–307, 1995.

[Ber87] F. Bergholm. Edge focusing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(6) :726–741, 1987.

[BFBB] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. Performance of optical flow techniques. *IEEE Conférence on Computer Vision and Pattern Recognition (CVPR)*, 92 :236–242.

[Bha43] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their population distributions. *Bulletin Calcutta Mathematical Society*, 35 :99–109, 1943.

[Bie85] I. Biederman. Human image understanding : Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32(1) :29–73, October 1985.

[Bor84] G. Borgefors. Distance transformations in arbitrary dimensions. *Comput. Vision Graph. Image Process.*, 27(3) :321–345, September 1984.

[Bor86] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3) :344–371, Jun 1986.

[BPB05] R.V. Babu, P. Perez, and P. Bouthemy. Robust tracking with motion estimation and kernel-based color modeling. *International Conference on Image Processing (ICIP05)*, 1 :717–720, 2005.

[BR02] L. Birgé and Y. Rozenholc. How many bins should be put in a regular histogram. Technical report, Laboratoire Probabilités et Modèles Aléatoires, Université Pierre et Marie Curie, Paris, France, PMA-721, 2002.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(6) :679–698, 1986.

[CG99] D. Crisan and M. Grunwald. Large deviation comparison of branching algorithms versus resampling algorithms : application to discrete time stochastic filtering. *Statist. Lab., Cambridge Univ., U.K., Tech. Rep., TR1999-9*, 1999.

[Che92] L.H. Chen. A two-phase area-level line/edge detector. *Pattern Recogn.*, 25(1) :55–63, 1992.

[CHTH01] T.F. Cootes, A. Hill, C.J. Taylor, and J. Haslam. Statistical models of appearance for medical image analysis and computer vision. *SPIE Medical Imaging*, 2001.

[CP95] J.P. Cocquerez and S. Philipp. *Analyse d'images : filtrage et segmentation.* Ed. Masson, 1995.

[CRM] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, pages 142–151, Hilton Head Island, South Carolina.

[CRM03] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(5) :564–577, May 2003.

[CS02] S.H. Cha and S.N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6) :1355–1370, June 2002.

[DDHF06] G. Dewaele, F. Devernay, R. Horaud, and F. Forbes. The alignment between 3-d data and articulated shapes with bending surfaces. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proceedings of the 9th European Conference on Computer Vision, Graz, Austria*, volume III of *LNCS*, pages 578–591. Springer, May 2006.

[DDJ02] F. Duculty, M. Dhome, and F. Jurie. Real time 3d face tracking from appearance. In *International Conference on Image Processing*, pages I 581–584, New-York, USA, September 2002.

[DGA00] A. Doucet, S. J. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3) :197–208, 2000.

[DH72] R.O. Duda and P.E. Hart. Use of the hough transform to detect lines and curves in pictures. *Communication of the Association for Computing Machinery (ACM)*, 15(1) :11–15, January 1972.

[dMRS92] P. del Moral, G. Rigal, and G. Salut. Estimation et commande optimale non-linéaire. *Rapport de Recherche du LAAS*, 2, 1992.

[dO94] P. Milheiro de Oliveira. Approximate filters for a nonlinear discrete time filtering problem with small observation noise. *Stochastics and Stochastics Reports*, 46(24), 1994.

[Doc01] S.L. Dockstader. Multiple camera tracking of interacting and occluded human motion. *Proc. of the IEEE*, 89(10) :1441–1455, 2001.

[Dor94] B. Dorner. *Chasing the color glove : Visual hand tracking.* PhD thesis, Simon Fraser University, 1994. Master Thesis.

[DT01] S.L. Dockstader and A.M. Tekalp. Multiple camera fusion for multi-object tracking. In *Proceedings of the IEEE Workshop on Multi-Object Tracking (WOMOT'01)*, page 95, Washington, DC, USA, 2001. IEEE Computer Society.

[Fau85] O.D Faugeras. *The Third International Symposium in Robotic Research.* MIT Press, 1985.

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Communication of the Association for Computing Machinery (ACM)*, 24(6) :381–395, June 1981.

[FG87] W. Forstner and E. Gulch. A fast operator for detection and precise location of distinct points, corners, and centers of circular features. In *In Proc. Intercommision Conference on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

[FL88] O.D. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence (PRAI)*, 2(3) :485–508, 1988.

[GB01] W.R. Gilks and C. Berzuini. Following a moving target. monte carlo inference for dynamic bayesian models. *Journal of the royal statistical society, Series B Statistical Methodology*, 63(1) :127–146, 2001.

[GC01] S. Godsill and T. Clapp. *Improvement strategies for Monte Carlo particle filters.* Springer-Verlag, 2001.

[Gel74] A. Gelb. *Applied Optimal Estimation.* MIT Press, Cambridge Mass, 1974.

[GSS93] N.J. Gordon, D.J. Salmond, and A.F. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Proceedings of IEE F : Image, Vision Signal Processing*, 140 :107–113, 1993.

[Han70] J.E. Handschin. Monte carlo techniques for prediction and filtering of non linear stochastic processes. *Automatica*, 6 :555–563, 1970.

[Har92] C.G. Harris. *Tracking with rigid models.* In A. Blake and A. Yuille editors, Active Vision, MIT Press, 1992.

[Har97] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(6) :580–593, Jun 1997.

[HH96] T. Heap and T.S. Huang. 3d deformable hand models. *Proceedings of Gesture Workshop'96*, pages 131–139, 1996. Univ. of York.

[HH98] T. Heap and D.C. Hogg. Wormholes in shape space : Tracking through discontinuous changes in shape. In *International Conference on Computer Vision (ICCV98), Bombay,India*, pages 344–349, 1998.

[HKRR81] L.D. Harmon, M.K. Khan, R.Lascha, and P.F. Raming. Machine identification of human faces. *Pattern Recognition*, 13(2) :90–110, 1981.

[HM93] R. Horaud and O. Monga. *Vision par ordinateur.* Hermès, 1993.

[Hou62] P.V.C. Hough. Method and means for recognizing complex patterns. In *US Patent*, 1962.

[HS81] B.K.P. Horn and B.G. Schunck. Determining optical flow. *AI*, 17(1-3) :185–203, August 1981.

[HS88] C. Harris and M.J. Stephens. A combined corner and edge detector. In *Alvey88 (4th Alvey Vision Conference)*, pages 147–152, 1988.

[HS90] C. Harris and C. Stennet. Rapid - a video rate object tracker. *In British Machine Vision Conference*, pages 73–78, 1990. Oxford.

[Hub81] P.J. Huber. *Robust Statistics.* Wiley, 1981.

[Hue03] C. Hue. *Méthodes séquentielles de Monte Carlo pour le filtrage non linéaire multi-objets dans un environnement bruité. Applications au pistage multi-cibles et à la trajectographie d'entités dans des séquences d'images 2D.* PhD thesis, Université de Rennes, 2003.

[HZ00] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, Jun 2000.

[IB96a] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Conference on Computer Vision (ECCV96)*, volume 1, pages 343–356, 1996.

[IB96b] M.A. Isard and A. Blake. Visual tracking by stochastic propagation of conditionnal density. *In Proceedings of the 4th European Conference on Computer Vision (ECCV)*, pages 343–356, 1996. Cambridge, England.

[IB98] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *International Conference on Computer Vision (ICCV98)*, pages 107–112, 1998.

[Jai81] R. Jain. Extraction of motion information from peripheral process. *IEEE Trans. Patt. Anal. Mach. Int.*, 3 :489–503, 1981.

[JC02] G. Jaffré and A. Crouzil. Utilisation de la procédure Mean Shift pour le problème du suivi de joueurs dans des séquences d'images d'activités sportives . Rapport de recherche 2002-33-R, IRIT, Université Paul Sabatier, Toulouse, octobre 2002.

[JD00] F. Jurie and M. Dhome. Un algorithme efficace de suivi d'objets dans des séquences d'images. *Ronnaissance des Formes et Intelligence Artificielle (RFIA)*, 2000.

[JD01] F. Jurie and M. Dhome. Real time 3d template matching. *IEEE Conférence on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[JD02] F. Jurie and M. Dhome. Real time robust template matching. In *British Machine Vision Conference*, pages 123–132, 2002.

[JFEM01] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust online appearance models for visual tracking. In *IEEE Conférence on Computer Vision and Pattern Recognition (CVPR01)*, pages I :415–422, 2001.

[Jol06] J.M. Jolion. Probabilités et statistiques. *Document de cours, INSA Lyon*, 2006.

[JSR05] A. Jacquot, P. Sturm, and O. Ruch. Adaptive tracking of non-rigid objects based on color histograms and automatic parameter selection. In *Proceedings of IEEE Workshop on Motion and Video Computing, Breckenridge, Colorado*, volume 2, pages 103–109. IEEE Computer Society, January 2005.

[JU97] S. Julier and J. Uhlmann. A new extension of the kalman filter to non linear systems. *International Symposium Aerospace/Defense Sensing, Simulations and Controls*, 1997.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering (T-ASME)*, 82 :35–45, March 1960.

[KB99] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. *In Proc. the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 85–94, 1999.

[KDN94] D. Koller, K. Daniilidis, and H.H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision (IJCV)*, 10 :257–281, 1994.

[KH95] J.J Kush and T.S. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. *Proceedings of the 5th International Conference on Computer Vision (ICCV)*, pages 666–671, 1995.

[Kit87] G. Kitagawa. Non-gaussian state space modeling of non stationary time series (with discussion). *Journal of the American Statistical Association*, 82 :1032–1041, 1987.

[Kit96] G. Kitagawa. Monte carlo filter and smoother for non-gaussian non linear state space models. *Journal of Computational and Graphical Statistics*, 5 :1–15, 1996.

[KL51] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22 :76–86, 1951.

[KLW94] A. Kong, J.S. Liu, and W.H. Wong. Sequential imputation and bayesian missing data problems. *Journal Americian Statistical Association*, pages 278–288, 1994.

[KR04] T. Korah and C. Rasmussen. Probabilistic contour extraction with model-switching for vehicle localization. *IEEE Intelligent Vehicle Symposium*, 4, 2004. Parma, Italy.

[Kun90] A. Kundu. Robust edge detection. *Pattern Recognition*, 23(5) :423–440, 1990.

[KWT87] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes : Active contour models. In *International Conference on Computer Vision (ICCV87)*, pages 259–268, 1987.

[Lan97] Z.D. Lan. *Méthodes robustes en vision : application aux appariement visuels.* PhD thesis, INPG, 1997.

[LC95] J.S. Liu and R. Chen. Blind deconvolution via sequential imputation. *Journal of the American Statistical Association*, 90 :567–576, 1995.

[Leg03] F. Legland. Filtrage particulaire. *Actes 19ème Colloque GRETSI sur le Traitement du Signal et des Images*, 1 :1–8, 2003.

[Leg05] F. Legland. Introduction au filtrage en temps discret : Filtre de kalman, filtrage particulaire, modèles de markov cachés. *Université de Rennes 1, Master Recherche STI*, 2005.

[LF96] Q.T. Luong and O.D. Faugeras. The fundamental matrix : Theory, algorithms, and stability analysis. *International Journal of Computer Vision (IJCV)*, 17(1) :43–75, January 1996.

[LF05] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects : A survey. *In Foundations and Trends in Computer Graphics and Vision*, 1(1) :1–89, 2005.

[LFMV01] S. Lefevre, C. Fluck, B. Maillard, and N. Vincent. Un modèle de contour actif pour le suivi rapide d'objets en mouvement. application au suivi de joueurs de football. *Colloque GRETSI sur le Traitement du Signal et des images*, pages 501–504, Septembre 2001. Toulouse, France.

[LH81] H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. pages 133–135, 1981.

[Lin91] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1) :145–151, 1991.

[Liu98] J.S. Liu. Sequential monte carlo methods for dynamic systems. *Journal American Statistical Association*, 93 :1032–1044, 1998.

[LK95] J. Lee and T.L. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, pages 77–86, 1995.

[LLLP05] A. Lehuger, P. Lechat, N. Laurent, and P. Pérez. Suivi de joueurs dans les séquences sportives à fort changement d'illumination : évaluation du problème et solutions. In *Proc. Journées Compression et Représentation des Signaux Visuels (CORESA'05)*, Rennes, France, November 2005.

[LV05] S. Lefevre and N. Vincent. Contours actifs pour le suivi d'objet en temps-réel : multi-topologies et multi-résolutions. *20e Colloque GRETSI*, pages 1125–1128, Septembre 2005. Belgique.

[LY95] M.K. Leung and Y.H. Yang. First sight : A human-body outline labeling system. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(4) :359–377, April 1995.

[Mat55] K. Matusita. Decision rules based on distance for problems of fit, two samples and estimation. *Annals of Mathematical Statistics*, 26 :631–641, 1955.

[MDFW00] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan. The unscented particle filter. Technical report, University of Cambridge, Engineering department, 2000.

[MMKR91] P. Meer, D. Mintz, D.Y. Kim, and A. Rosenfeld. Robust regression methods for computer vision : A review. *International Journal of Computer Vision (IJCV)*, 6(1) :59–70, April 1991.

[MOG01] C. Musso, N. Oudjane, and F. Le Gland. *Improving regularised particle filters.* Springer-Verlag, 2001.

[Mor80] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University*, September 1980. Available as Stanford AIM-340, CS-80-813 and republished as a Carnegie Mellon University Robotics Institue Technical Report to increase availability.

[NC96] U. Neumann and Y. Cho. A self-traking augmented reality system. *In Proc. VRST 96*, pages 109–115, 1996.

[Nea01] R. Neal. Annealed importance sampling. *Statistics and Computing*, 11 :125–139, 2001.

[NKMG02] K. Nummiaro, E. Koller-Meier, and L.J. Van Gool. Object tracking with an adaptive color-based particle filter. In *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, pages 353–360, London, UK, 2002. Springer-Verlag.

[NKMG03] K. Nummiaro, E. Koller-Meier, and L. Van Gool. Color features for tracking non-rigid objects. *Special Issue on Visual Surveillance, Chinese Journal of Automation*, 29(3) :345–355, 2003.

[NS04] H.T. Nguyen and A.W.M. Smeulders. Fast occluded object tracking by a robust appearance filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(8) :1099–1104, August 2004.

[PD99] N. Paragios and R. Deriche. Geodesic active regions for motion estimation and tracking. In *International Conference on Computer Vision (ICCV99)*, pages 688–694, 1999.

[PF97] D. Terzopoulos P. Fieguth. Color based tracking of heads and other mobile objects at video frame rates. *In Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1997. Puerto Rico.

[PFTV93] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1993.

[PHVG02] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. *Conference on Computer Vision (ECCV)*, 1 :661–675, 2002.

[Pic91] J. Picard. Efficiency of the extended kalman filter for non linear systems with small noise. *Journal on Applied Mathematics*, 51(3) :843–855, 1991.

[PM05] M. Presigout and E. Marchand. A model free hybrid algorithm for real time tracking. *IEEE International Conference on Image Processing (ICIP'05)*, 3 :97–100, September 2005. Gène, Italie.

[PP04] A. Blake P. Perez, J. Vermaak. Data fusion for visual tracking with particles. *Proceedings of the IEEE (issue on State Estimation)*, 2004.

[PS99] M.K. Pitt and N. Shephard. Filtering via simulation : auxiliary particle filters. *Journal of the American Statistical Association*, 94 :590–599, 1999.

[PS01] M. Pitt and N. Shepard. *Auxiliary variable based particle filters.* Springer-Verlag, 2001.

[RC98] L. Reveret and L. Le Chevalier. Un modèle géométrique de lèvres 3d adaptable au locuteur. *Actes des XXIIemes Journees d'Etudes sur la Parole, JEP'98*, pages 213–216, Juin 1998. Martigny, Suisse.

[RJ86] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov Models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

[RK94] J.M. Rehg and T. Kanade. Digiteyes : Vision-based hand tracking for human-computer interaction. *In Proceedings of the workshop on Motion of Non-Rigid and Articulated Bodies*, pages 16–24, November 1994.

[RL87] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection.* John Wiley & Sons, Inc., New York, NY, USA, 1987.

[Roh94] K. Rohr. Toward model-based recognition of movement in image sequences. *Computer Vision, Graphique and image processing : image understanding*, 59(1) :94–115, 1994.

[RP66] A. Rosenfeld and J.L. Pfaltz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery (ACM)*, 13(4) :471–494, October 1966.

[SA71] H.W. Sorenson and D.L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatica*, 7 :465–479, 1971.

[SB95] S.M. Smith and J.M. Brady. SUSAN – A new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995.

[Sch96] C. Schmid. *Appariement d'images par invariants locaux de niveaux de gris.* PhD thesis, INPG, Juillet 1996.

[Sen02] A. Senior. Tracking people with probabilistic appearance models. In *Performance Evaluation of Traking Systems (PETS02)*, pages 48–55, 2002.

[SHE98] A. Schödl, A. Haro, and I.A. Essa. Head tracking using a textured polygonal model. *Proc. Workshop Perceptual User Interfaces*, 1998.

[SHT+01] A. Senior, A. Hampapur, Y.L. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. In *Performance Evaluation of Traking Systems (PETS01)*, 2001.

[Sor88] H.W. Sorenson. *Recursive estimation for nonlinear dynamic systems.* ed. J. C. Spall, New York : Marcel Dekker, In Bayesian Analysis of Time Series and Dynamic Models, 1988.

[ST94] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[Tho80] W.B. Thomson. Combining motion and contrast for segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2 :543–549, 1980.

[WNC04] J. Wang, P. Neskovic, and L.N. Cooper. Tracking object features using dynamically define spatial context. *Early Cognitive Vision Workshop*, June 2004. Scotland.

[YLS04] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(11) :1531–1536, November 2004.

[YP05] M. Yokoama and T. Poggio. A contour-based moving object detection and tracking. *International Conference on Computer Vision (ICCV)*, 2005.

[ZK04] Z. Zivkovic and B. Krose. An em-like algorithm for color-histogram-based object tracking. In *IEEE Conférence on Computer Vision and Pattern Recognition (CVPR04)*, pages I : 798–803, 2004.

Résumé

Cette thèse résulte d'une collaboration CIFRE avec Thalès Optronique, société qui conçoit et développe des systèmes de désignation aéroportés pour des « armements guidés laser ». Le mode opératoire utilisé par ces systèmes requiert de la part du pilote une charge de travail importante. L'objectif de cette thèse est d'alléger cette charge, en rendant les systèmes de suivi actuels plus robustes face à des effets de perspective.

Le thème principal de cette thèse est donc le suivi d'objets à partir d'images aériennes. Nous souhaitons utiliser la faisabilité d'une extraction d'information 3D à partir de séquences vidéo afin d'améliorer les algorithmes de suivi de matériels aéroportés existants. Pour cela, nous nous plaçons dans un cadre bayésien et formulons le suivi de manière probabiliste, au moyen d'un filtre particulaire. Nous avons mis en place trois algorithmes : Le premier est basé sur des histogrammes de couleurs, que l'on combine à un filtrage particulaire ; c'est un suivi purement 2D dans le sens où aucune information 3D réelle de la scène est utilisée. Le second est basé sur des modèles géométriques (qui peuvent être 2D ou 3D), que l'on combine à un filtrage particulaire. Nous ajoutons une étape supplémentaire au filtrage particulaire classique, nous permettant de changer de modèle lorsque l'algorithme le juge nécessaire. Enfin, le dernier algorithme combine les deux précédents ; l'intégration d'histogrammes de couleurs et d'informations de contours dans un filtre particulaire permet non seulement de rendre le suivi d'objets plus robuste, mais aussi de prendre en compte de l'information 3D réelle de la scène observée.

Un protocole d'évaluation a été mis en place pour évaluer les performances de ces algorithmes. Des résultats illustrent les performances de ces algorithmes.

Abstract

This thesis results from a collaboration CIFRE with Thalès Optronique, a company which develops airborne systems for « laser guided armaments ». The operating mode used by these systems requires on behalf of the pilot an important workload. The objective of this thesis is to reduce this responsibility, by making the current trackers more robust to the three dimensional effects, in particular in the presence of reliefs.

The main theme of this thesis is the tracking of objects from aerial images. We want to extract some three dimensional information from video sequences in order to improve the existing algorithms. For that purpose, we place ourselves in a Bayesian framework and formulate the tracking problem by means of a particle filter. We set up three algorithms : The first one is based on color histograms, which we integrate in a particle filter ; The second one is based on geometrical models (that can be 2D or 3D), that we combine with a particle filter. We add a supplementary stage to the classical particle filter, allowing us to change model when the algorithm judges it necessary. And the last one combines both precedents ; the integration of histograms and contour information in a particle filter allows not only to have a more robust tracker, but also to take into account of the 3D real information of the observed scene.

An evaluation protocol has been set up to estimate the performances of these algorithms. Results illustrate the performances of these algorithms.